Spring 1995

A defect detection algorithm for sequential and parallel computers

Ralf Hross
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

A DEFECT DETECTION ALGORITHM FOR SEQUENTIAL AND PARALLEL COMPUTERS

by

Ralf Hross

The comparison of images containing a single object of interest, where one of them contains a model object, is frequently used for defect detection/identification. This is often a problem of interest to industrial applications. To make this task more efficient, the implementation on parallel computers should be a major objective. This thesis introduces sequential and parallel versions of an algorithm that compares original(reference) and processed images in the time and frequency domains. The emphasis is on parallel implementation. This comparison may help to detect changes in images. Extracted data is also compared to database data in an attempt to pinpoint specific changes, such as rotations, translations, defects, etc. However, our emphasis is on defect detection. The first application considered here is recognition of an object which has been translated and/or rotated. For illustration purposes, an original image of a centered hypodermic needle is compared to a second image of the needle in a different position. This algorithm will determine if both images contain the same object regardless of position. The second application detects changes(defects) in the needle regardless of position and reports the quality of the needle. This quality is reported with a quantitative measurement. Finally, the performance of sequential and parallel versions of the algorithm on a Sun SPARCstation and on an experimental in-house built parallel DSP computer with eight TMS320C40 respectively processors is included. The results show that significant speedup can be achieved through incorporation of parallel processing techniques.

A DEFECT DETECTION ALGORITHM FOR SEQUENTIAL AND PARALLEL COMPUTERS

by
Ralf Hross

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

May 1995

APPROVAL PAGE

A DEFECT DETECTION ALGORITHM FOR SEQUENTIAL AND PARALLEL COMPUTERS

by
Ralf Hross

Dr. Sotirios Ziavras, Thesis Advisor — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Constantine N. Manikopoulos, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Y-Q Shi, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Ralf Hross

Degree: Master of Science in Electrical Engineering

Date: May 1995

Undergraduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, New Jersey, 1995
- Bachelor of Science in Biomedical Engineering,
  University of Bridgeport, Bridgeport, Connecticut, 1988
- Bachelor of Science in Electrical Engineering,
  University of Bridgeport, Bridgeport, Connecticut, 1986

Major: Electrical Engineering

Presentations and Publications:

Lawrence V. Hmurcik, Ralf Hross, Doron Koren and Ron Rivilin, "Using Computer Simulation to Help Students Become More Independent in Their Thinking and Designing", 69th Annual Conference of the American Society for Engineering Education, University of New Haven, Connecticut, October 19, 1991.

Ralf Hross and Seema Winsor, "Low Frequency Spectrum Analyzer, Digital Signal Processing Application", University of Bridgeport Digital Signal Processing Lab Research, Spring 1991.

Ralf Hross, Louis Prastitis, Sotirios Ziavras, "Signal Analysis with a Newly Designed Electrocardiograph Using Switched Capacitor Filter", Proceedings of 19th IEEE Northeast Bioengineering Conference, Newark, New Jersey, Spring 1993.

Ralf Hross and Meng Chu Zhou, "Computer Integrated Radio Monitoring For Emergency Situations", 1994 IEEE International Conference on Systems, Man, and Cybernetics, October 1994.

Ralf Hross, Sotirios G. Ziavras, Constantine N. Manikopoulos, Nitin J. Lad, and Xi Li, "A Defect Identification Algorithm for Sequential and Parallel Computers", IEEE International Symposium on Industrial Electronics, Athens, Greece, July 10-14, 1995.

This thesis is dedicated to Ernest, Lore, Eva ,friends
and to the memory of Frank.

ACKNOWLEDGMENT

I would like to thank all the professors where classes I have taken in the last two years. Special thanks to Professors Ziavras and Manikopoulos for their guidance and advice, and to Y-Q Shi for his discussion in the area of image processing.

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION	1
2	THE TURBONET SYSTEM	3
	2.1 System Description	3
	2.2 The Host System	6
	2.3 VME Bus	6
	2.4 The TMS320C40 Digital Signal Processor	8
3	THE DEFECT DETECTION ALGORITHM	11
4	MATLAB SIMULATION	22
5	PARALLEL IMPLEMENTATION AND TIMING	25
6	CONCLUSION	27
APPENDIX A	IMAGES AND FIGURES	28
APPENDIX B	MATLAB PROGRAM CODE	34
APPENDIX C	SEQUENTIAL AND PARALLEL PROGRAM CODE FOR THE TURBONET INCLUDING MEMORY MAPPING	115
REFERENCES		196

LIST OF TABLES

Table		Page
1	Image Error Analysis of a Translated/Rotated Needle with/without Defects.....	18
2	FFT Error Analysis of a Translated/Rotated Needle with/without Defects.....	18
3	Sequential and Parallel Execution Time on TurboNet.....	25
4	File and Variable Name Coding.....	35
5	Memory Mapping.....	116

LIST OF FIGURES

Figure		Page
1.1	Test Flowchart of the General Algorithm	3
2.1	The NJIT TurboNet System	4
2.2	Block Diagram of Hydra Board	5
2.3	The TMS320C40 Block Diagram	9
2.4	The TMS320C40 Block Diagram	10
3.1	Flowchart of Defect Identification Algorithm	11
3.2	Image of Needle with no Defects Rotated and Translated	20
3.3	Analysis of Needle with no Defects Rotated and Translated	21
4.1	3D Intensity Plot of Needle Without Defects (time domain), Rotated and Translated	23
4.2	3D Intensity Plot of Needle Without Defects (frequency domain), Rotated and Translated	24
5.1	Parallel Flowchart of The Defect Detection Algorithm	26
A.1	Image of Needle with Defect (one pixel subtracted at the top), Rotated and Translated	28
A.2	Analysis of Needle with Defect (one pixel subtracted at the top), Rotated and Translated	29
A.3	Image of Needle with Defect (one pixel subtracted at the center), Rotated and Translated	30
A.4	Analysis of Needle with Defect (one pixel subtracted at the center), Rotated and Translated	31
A.5	Image of Needle with Defect (two pixels subtracted at the top), Rotated and Translated	32

LIST OF FIGURES

Figure Page

A.6 Analysis of Needle with Defect (two pixels subtracted at the top), Rotated and Translated 33

B.1 Program Flowchart 34

C.1 Block Diagram of Hydra 115

CHAPTER 1

INTRODUCTION

The processing of images at different levels has become a very important application in the military, industrial, commercial, medical, and meteorological areas. Image processing uses large amounts of data and requires significant processing time. With the development of new computer technologies, advanced complex image processing algorithms can often be executed in a reasonable amount of time. Parallel processing, an emerging modern computer technology, has been introduced as a tool for solving demanding problems, such as real time image processing.

A common problem in image processing involves the identification of changes that occurred to an original image [1]. Using this image as a reference, a transformed image is analyzed to pinpoint and describe the transformation process it went through. This kind of a process can belong to one of two categories: the content of the image has changed (example: an object was moved to a new position), or the entire image was processed (example: the brightness of the picture was increased).

In order to identify processes for these two categories, an algorithm that uses known image processing routines as a basis is introduced. Sequential and parallel versions of the algorithm are presented. This new algorithm first will be tested for a translated object. The object, a hypodermic needle, will be identified by the algorithm. A second application of this algorithm involves the identification of defects, again using the needle as an example. In the latter case, this object will be simulated on computers at

different resolutions and using only binary values. Also, UNIX, a well known operating system, and MATLAB, a mathematical matrix-based and graphics application software, are combined with customized C-based algorithms into an easy to use advanced image processing tool for sequential and customized parallel systems. The following diagram shows the general flowchart for testing the defect identification algorithm in the MATLAB environment (simulation on a SPARC-based computer only) and on TurboNet (practical implementation of the algorithm), and the interaction between these environments. Image X (original image) and Image Y (processed image) are simulated. Detailed explanation of each flowchart is provided in the following chapters.

CHAPTER 2

THE TURBONET SYSTEM

2.1 System Description

The in-house built TurboNet Parallel system [2] comprises a VME backplane, a SPARC CPU-2CE host board [3], two Hydra boards [4], two hard disk drives, a floppy drive, a CD-ROM, a VME bus logic analyzer, and a set of PC-ATs as depicted in Figure 2.1. There are four links between the two Hydra boards, each of them connecting two TMS320C40s [5] together in two different boards. Each Hydra board contains four C40s.

Each C40 has six communication ports. Three of them are used within the board to implement fully connected system of four C40s. A fourth port is used for an interboard connection. All eight processors form a three dimensional hypercube.

The Hydra, as shown in Figure 2.2, is a single-slot VME-based multiprocessor system containing four C40 chips. The VME bus and the global bus of each C40 are connected to an Internal Shared Bus (ISB). The ISB provides access to DRAM memory and other shared resources. Because of the DRAM memory and the hypercube architecture, the TurboNet system implements both the shared-memory and message-passing communication paradigms. This is one of the features that make this system unique.

The TurboNet system is monitored by two PC-AT computers. These units are linked to the VME and Hydra boards. Their purpose is to display, using customized software, the VME status of the system and the Hydra board activity. Any error in the system will alert the programmer for troubleshooting and maintenance purposes

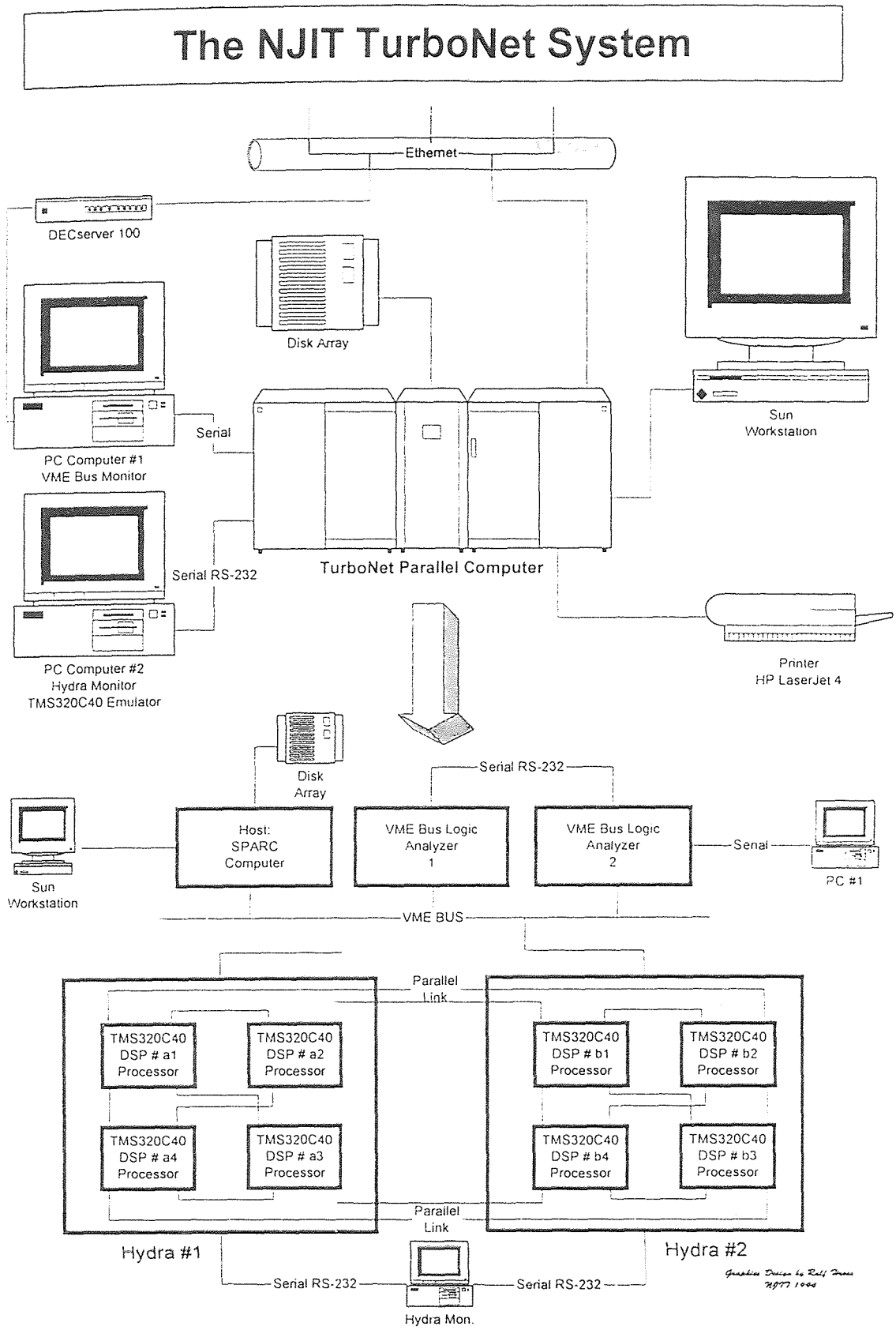

Figure 2.1 The NJIT TurboNet System

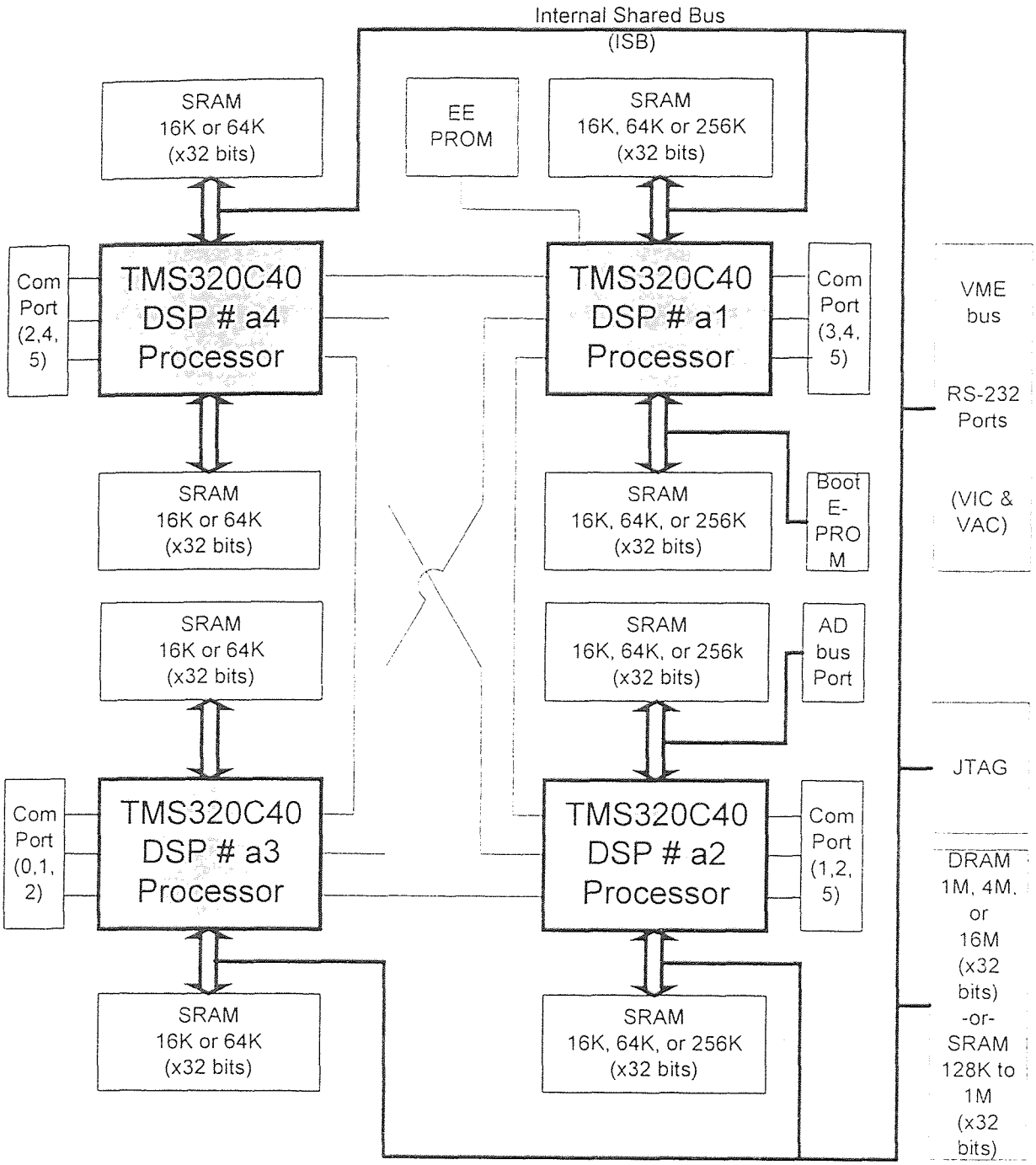

Figure 2.2 Block Diagram of Hydra Board.

2.2 The Host System

The host system, a SPARC CPU-2CE board, is a complete VME-based SPARC station 2 architecture with Sbus expansion. The system includes DMA-supported SCSI and Ethernet ports along with audio, keyboard/mouse, and two serial channels with modem support. The SPARC CPU-2CE runs the SunOS/Solaris operating system. It contains a 40 MHz SPARC 32-bit RISC processor with an integrated Integer/Floating-Point Unit, a Sun standard SRAM-based memory management unit, a cache controller, and two Cache RAM chips.

Operating at 40 MHz, the Integer/Floating Point Unit provides 28.5 MIPS integer performance and 4.2 MFLOPS floating point performance. The purpose of the host system is to compile and download the C40 programs to the Hydra boards using the VME bus.

2.3 VME Bus

The Hydra (figure 2.2) contains a fully functional VME bus Master/Slave with the following features:

- Supports VME Read, Write and Block transfers as a master or slave.
- Supports D8, D16 and D32 cycles as well as A16, A24 and A32 cycles.
- On-board VMEbus DMA controller

In addition, the Hydra implements VME Slot 1 system controller capabilities as follows: Built in VMEbus Arbiter supports single-level, priority, and round robin arbitration VME Interrupt support. Complete support for VME Interrupts: Interrupter and Interrupt Handler.

All DSPs have the capability of becoming the VME bus Master as well as the VME System controller. The Hydra VME interface has a built in DMA controller that can be set to move data to/from the shared internal DRAM, from to another VME card autonomously. This helps to relieve the DSPs from the task of data movement. The VME bus include up to 64-bit address and data buses, multiprocessing capability and seven level interrupt protocol. Both the address and data busses can be dynamically configured. This allows system expansion as microcomputer technology grows. VMEbus uses a master-slave architecture. Master modules transfer data to and from slave modules. Since many masters can reside on the bus it is called a multiprocessing bus. Before a master can transfer data it must first acquire the bus using a central arbiter, which is part of the system controller. Its function is to determine which master gets access to the bus.

VMEbus analyzers are logic analyzers that are designed specifically to interface and troubleshoot the VMEbus. The VMEbus testing can be done at the software and hardware level. Hardware handshaking and timing problems can be traced, analyzed and displayed on an independent terminal without interference to the VME bus. An example of such a display format is shown on the next page. The analyzer used for monitoring data transfer between the Hydras and the host is manufactured by VMETRO.

2.4 The TMS320C40 Digital Signal Processor

The TMS320C40 is a 32-bit processor designed specifically for parallel processing, to support real time embedded applications. Six communications ports for high-speed interprocessor communications having a 20-Mbyte/sec maximum asynchronous transfer rate are designed into the chip. The ports are driven by a six-channel DMA coprocessor for concurrent I/O and CPU operation, thereby maximizing sustained CPU performance by alleviating the CPU of burdensome I/O. The DSP CPU is capable of 275 MOPS and 320 Mbytes/sec. Two identical external data and address buses can support shared memory systems, and high data rate, single-cycle transfer is also designed into the chip. The total memory reach of the TMS320C40 is 4 billion 32-bit words. Program memory, including on chip RAM, ROM and external memory are contained within this area. This allows tables, coefficients, program code, and data to be stored locally. See Figure 2.3 and 2.4 for detailed view of the internal layout of the TMS320C40.

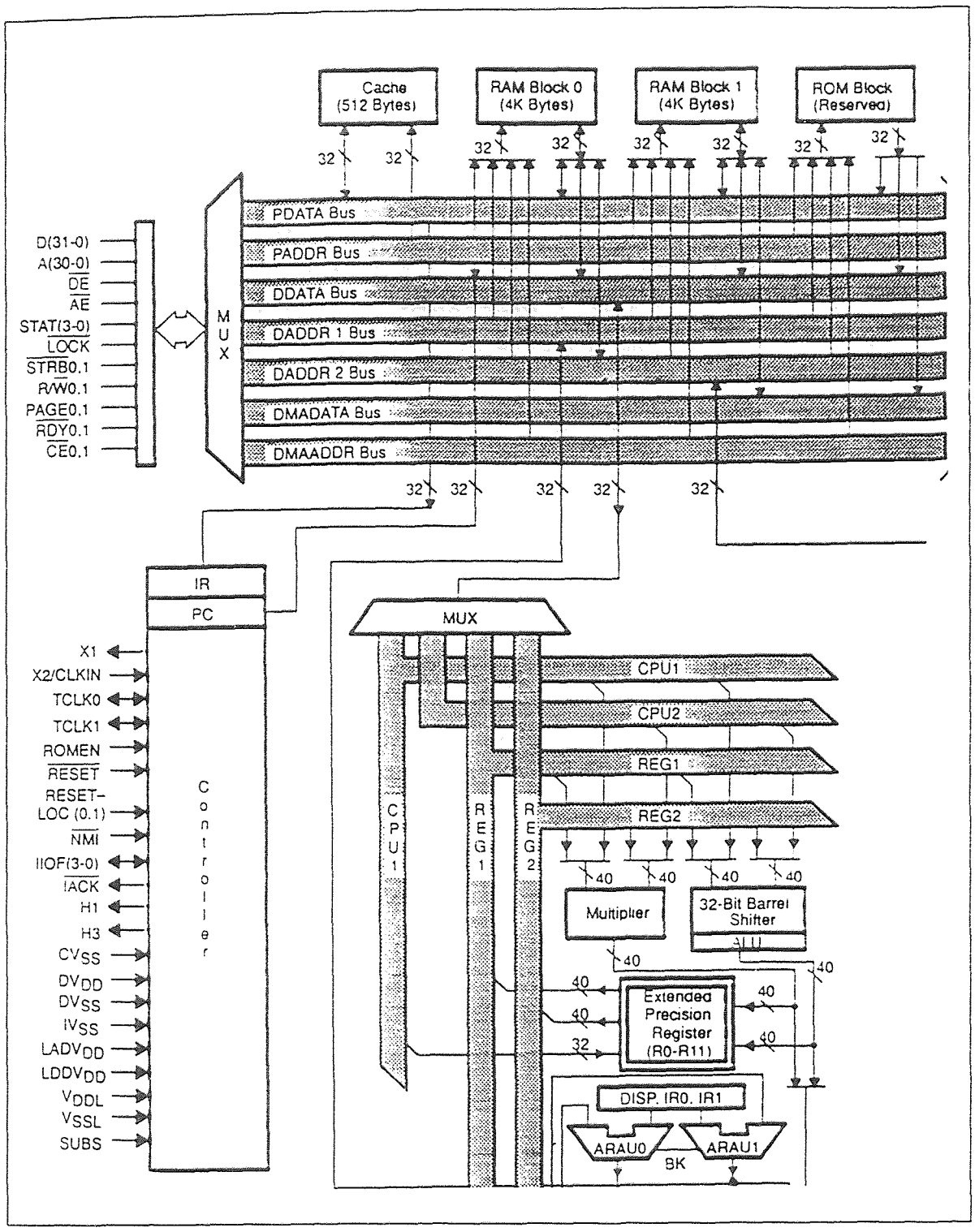

Figure 2.3 The TMS320C40 Block Diagram

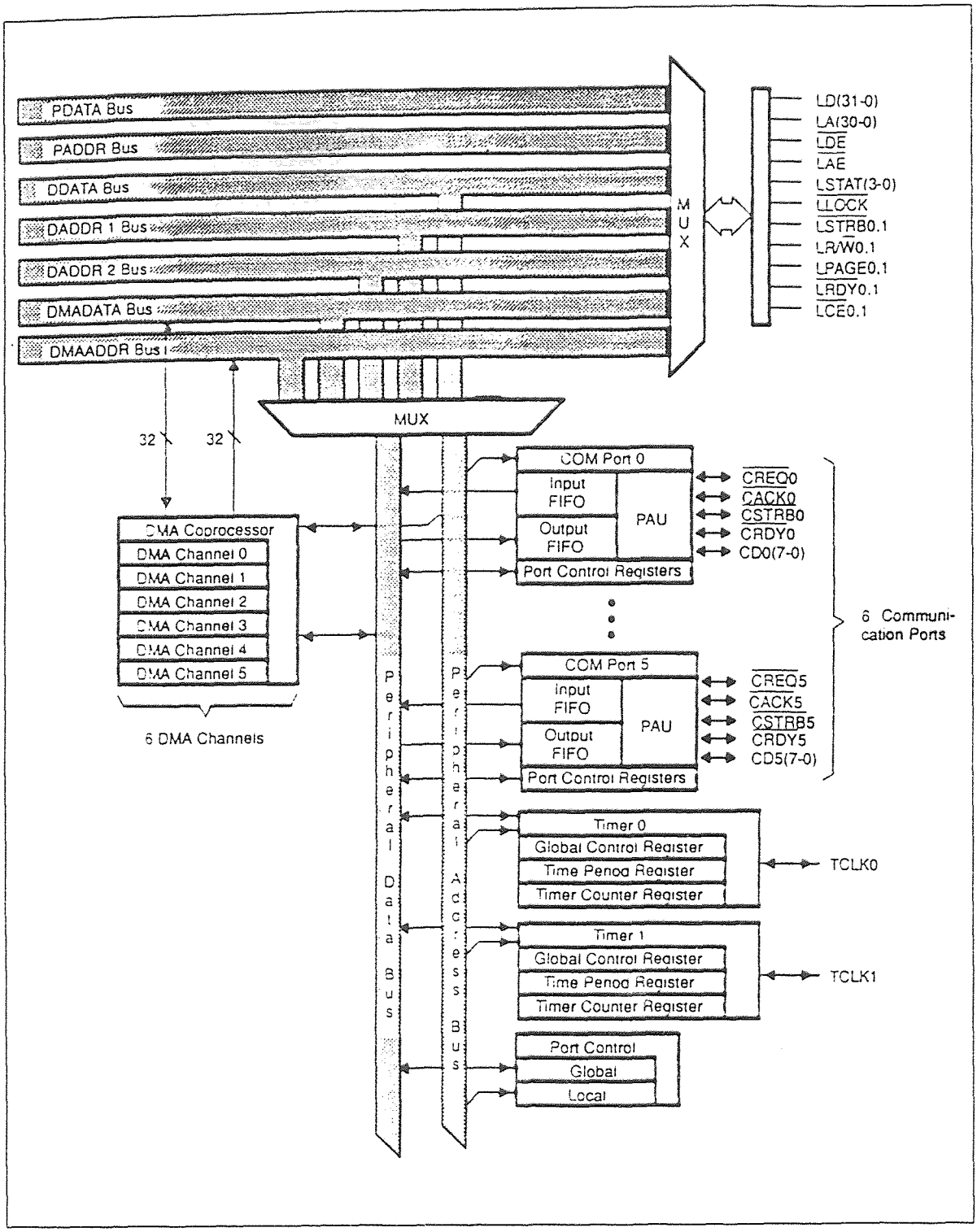

Figure 2.4 The TMS320C40 Block Diagram

CHAPTER 3

THE DEFECT DETECTION ALGORITHM

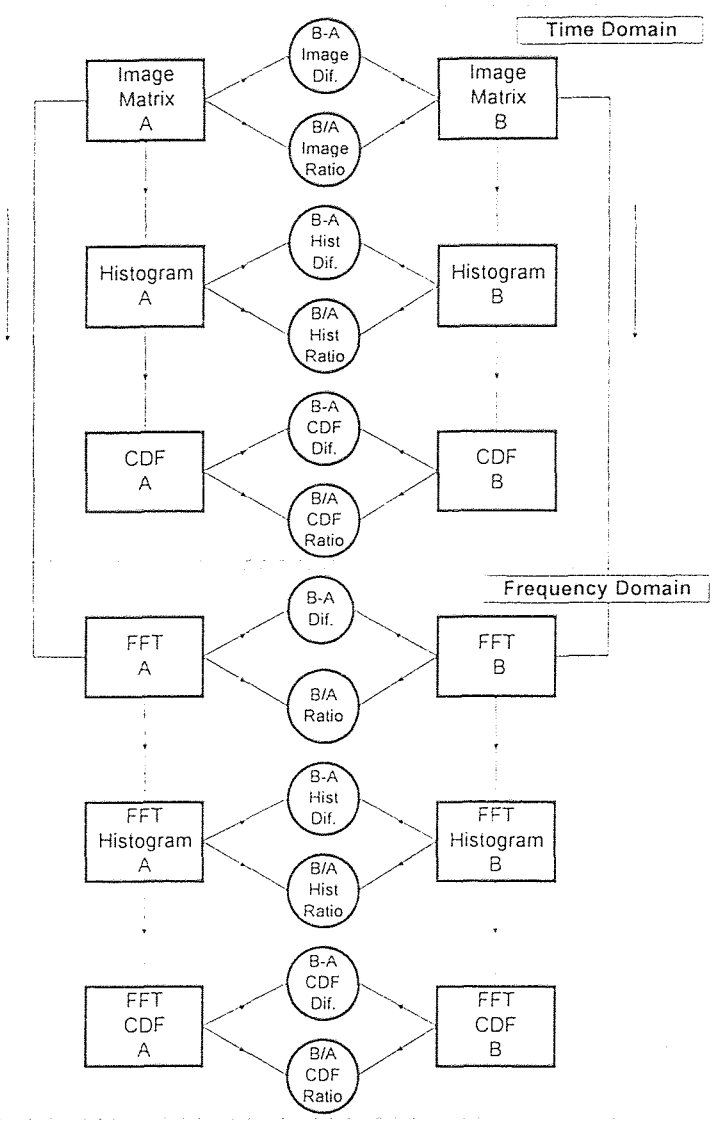

Figure 3.1 Flowchart of Defect Identification Algorithm

Using common mathematical tools for image processing, as Histograms [1] (gray level count) and the cumulative distribution function (CDF) [1] in both the time and frequency domains, a useful multipurpose algorithm can be formed which can give a lot of useful

information about a given picture. The algorithm considered here (Figure 3.1) takes two given pictures, finds their FFTs and compares them using histograms and CDFs, in both the time and frequency domains. One application for this algorithm is pattern recognition, where a model image is compared with another one for object recognition purposes. An error analysis is also included here for decision making. The input/output specifications and the processes applied follow.

-*Input image*: The original image, a 32x32 matrix of 64 gray levels, ranging from 0 (black) to 63 (white). Only two gray levels are used in this picture ,0 and 63, after thresholding

-*Output image:* The processed image, in this case rotated 90 degrees and translated, keeping the same size and color. Also, a defect can exist (addition or subtraction of pixels).

-*Two dimensional DFT:* Both images (input and output) will be mapped in the frequency domain using the two dimensional DFT (2d DFT) as follows:

The two variable discrete Fourier transform [1] of pixel values *f(x,y)* is defined as :

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-j2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)\right]$$

where u, v = 0, 1, 2,..... , N - 1. The Fourier spectrum is then obtained from the magnitude of each complex term. Due to large dynamic values rescaling will be necessary using the following equation :

$$D(u,v) = c \times \log\left[1 + \left|F(u,v)\right|\right]$$

This technique compensates for the difficulty of displaying the absolute value directly. This allows a visible increase in detail. A second scaling takes the previous values and maps them to the 0 to 63 gray level range. This mathematical process will round the numbers off, leaving only integers from 0 to 63. Discrete fourier analysis produces the same spectrum if the pictured object, in this case the needle, is translated or rotated 90 degrees. The following example explains this principle. Assume a 4 x 4 original matrix a, where the 1s represent the object, and a matrix b, where the object is translated and rotated.

$$a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculating the 2-dimensional discrete fourier transforms and their magnitude on a and b results in:

$$|dft2d(a)| = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix} \quad |dft2d(b)| = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The values were not divided by N. From these results we conclude that a shift in $f(x,y)$ does not affect the *magnitude* of its Fourier transform. This property is found in the following correspondence between the function and its Fourier transform [1]:

$$f(x-x_0, y-y_0) \Leftrightarrow F(u,v)\exp[-j2\pi(ux_0 + vy_0)/N]$$

and its magnitude property

$$\left|F(u,v)\exp[-j2\pi(ux_0 + vy_0)/N]\right| = |F(u,v)|.$$

Rotating $f(x,y)$ by an angle θ rotates $F(u,v)$ by the same angle. The previous 90 degree matrix rotation and translation illustrates this property: Rotation of $f(x,y)$ may create a distortion of the original picture, due to sampling. This distortion is defined as the addition or subtraction of pixel values. One pixel rotated may be averaged out as two pixels. The effect of such distortion is resolution dependent. The higher the resolution (matrix size), the lower the distortion. The following example illustrates a 45 degree nondistorting rotational property:

$$c = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad |dft2d(c)| = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

where *a* is the original and *c* is the processed (rotated image).

-*Histograms:* A gray level count of the image or matrix. Using the previous example, the matrices a, b, and c have the following gray level count :

$$histogram(a,b,c) = [12 \quad 4 \quad 0 \quad 0 \ldots\ldots\ldots]$$
$$gray\,values\ldots\ldots\ldots = [0 \quad 1 \quad 2 \quad 3 \ldots\ldots\ldots 63]$$

where histogram(a) = histogram(b) =histogram(c) and the two dimensional discrete Fourier transforms of the matrices a, b, c and c, have the following gray level count :

$$histogram(a,b,c) = [12 \quad 0 \quad 0 \quad 0 \quad 4 \ldots\ldots]$$
$$gray\,values\ldots\ldots\ldots = [0 \quad 1 \quad 2 \quad 3 \quad 4 \ldots\ldots 63]$$

From the histogram values obtained in the time domain *f(x,y)* and frequency domain *F(u,v)*, regardless of object translation and non-distortional rotation, all the gray level counts in the time domain for a, b, and c are equal to the ones in the frequency domain. This principle is applied for pattern recognition. In this paper a needle is simulated as a

line, and it is translated and rotated for demonstration purposes. Independent of position, a defective needle should be identified. This defect should be classified for reference purposes and decision making.

-Difference and Ratio: Comparison of the input and output images in any domain. Used for error analysis and identification of local or global changes. Using the previous histogram analysis, the difference and ratio of images a and b are equal to:

$$(b-a)=\begin{bmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \quad (b/a)=\begin{bmatrix} 1 & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \end{bmatrix}$$

The same is applied to the histogram results. The difference histogram between a and b in both the time and frequency domains is equal to

$$histogram(a,b)=[0 \quad 0 \quad 0 \quad 0 \quad 0.....0]$$
$$gray\ values.....=[0 \quad 1 \quad 2 \quad 3 \quad 4.....63]$$

This concludes that zero histogram differences in the time and frequency domains have to be the same object, regardless of non distortional rotation and translation. The algorithm has identified the same object (i.e. the needle) in the original and processed images. To simplify this detection a further step is introduced using error analysis.

-Error analysis: used for numerical interpretation of the defect of an object. The formulae known as the root square-error and mean square signal to noise ratio are used. They are formulated as follows:

$$e_{rms} = \sqrt{\left[\frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left[f_{original}(x,y) - f_{processed}(x,y) \right]^2 \right]}$$

$$SNR_{rms} = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f_{processed}(x,y)^2}{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left[f_{processed}(x,y) - f_{original}(x,y) \right]^2}$$

These two equations are used to numerically generate the following data of an original picture compared to a rotated/ translated. Three different defect conditions are of interest here:

- *Tr/Ro:* Needle with no defects is rotated and translated (Figure 4a, image_y).
- *1pixel:* Needle is rotated and translated with one pixel subtracted at the top (Figure 4b, image_y).

- *1pixel_s:* Needle is rotated and translated with one pixel subtracted at any position.
- *2pixel:* Needle is rotated and translated with two pixel subtracted at the top.

Results for 32x32 images are shown on Tables 1 and 2.

Table 1 Image Error Analysis of a Translated/Rotated Needle with /without Defects

	Image Error	Image SNR	Image Hist. Error	Image Hist. SNR	Image CDF Error	Image CDF SNR	Figure Locator
Tr/Ro	21.56	0.55	0.00	∞	0	∞	fig 3.1 image_y
1pixel	21.47	0.54	0.25	461953	1.40	9.3E5	fig A.1 image_y
1pixel s	21.47	0.54	0.25	461953	1.40	9.3E5	fig A.3 image_y
2 pixels	21.38	0.54	0.5	115712	2.8	2.3E5	fig A.5 image_y

Table 2 FFT Error Analysis of a Translated/Rotated Needle with /without Defects

	FFT Error	FFT SNR	FFT Hist. Error	FFT Hist. SNR	FFT CDF Error	FFT CDF SNR	Figure Locator
Tr/Ro	4.22	25.42	0	∞	0	∞	fig 3.2 fft image_y
1pixel	3.99	28.9	16.1	14.9	17.03	4.6E3	fig A.1 fft image_y
1pixel s	4.00	29.59	19.18	11.69	36.46	1.00E3	fig A.3 fft image_y
2pixel	4.50	22.3	14.67	15.03	17.67	8.3E3	fig A.5 fft image_y

From these results we can see that, in the time domain, the histogram difference shows the amount of error present. The frequency domain is more sensitive to position location, as shown in the one pixel example. For pattern recognition, both domains are necessary

for object confirmation, since two different objects in the time domain may produce the same histogram information. The frequency domain will confirm the shape, regardless of non-distortional translation/rotation.

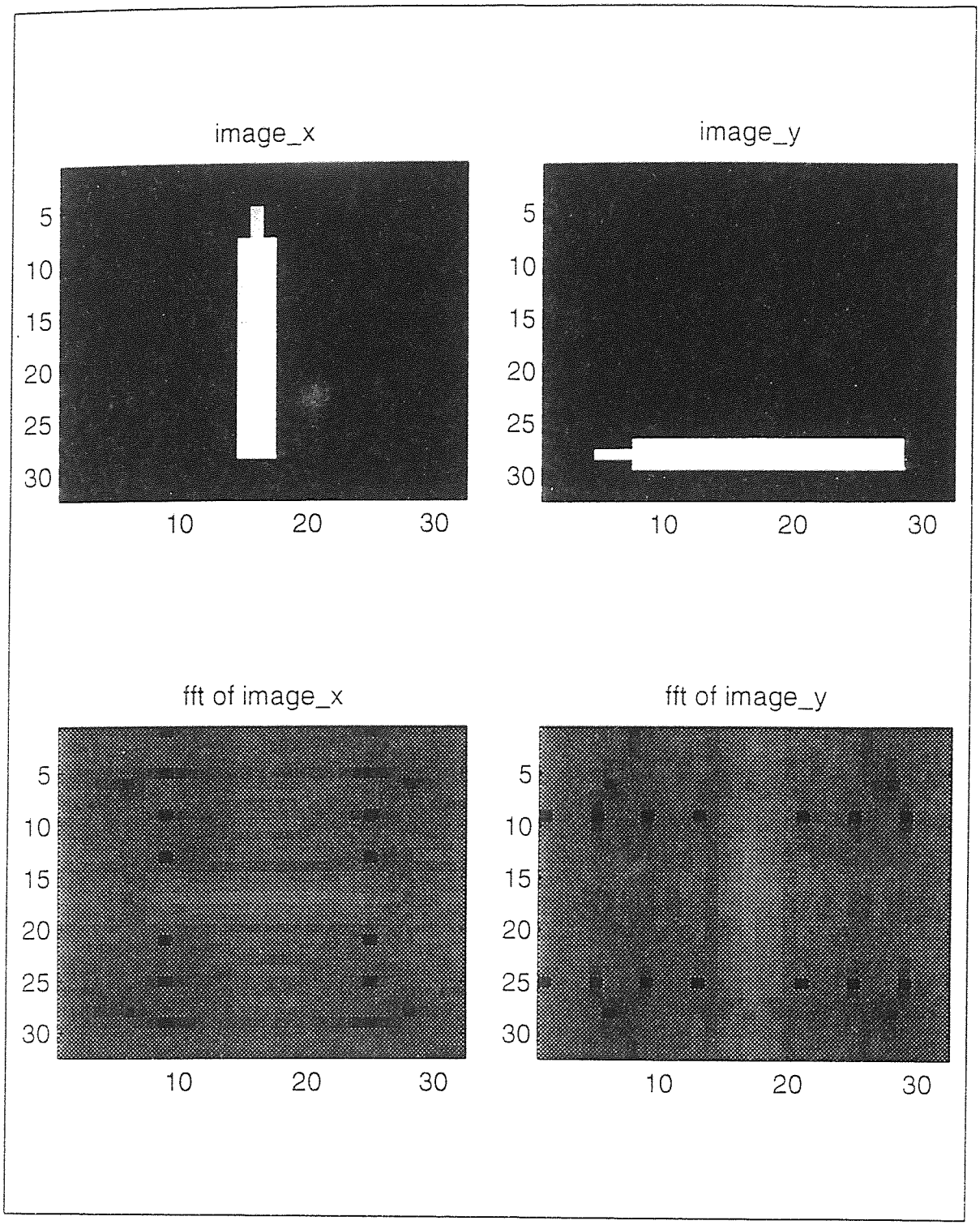

Figure 3.1 Image of Needle with no Defects Rotated and Translated

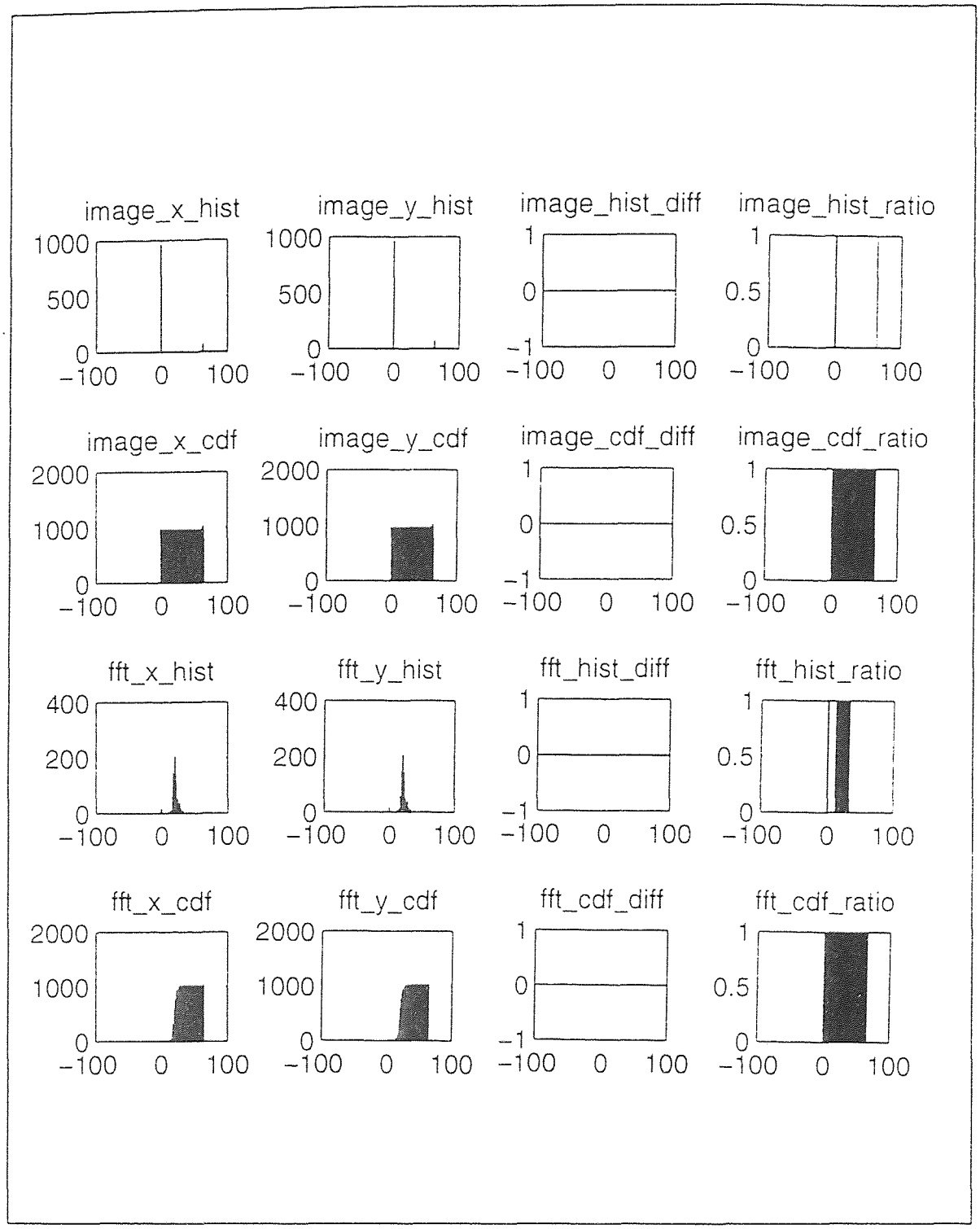

Figure 3.2 Analysis of Needle with no Defects Rotated and Translated

CHAPTER 4

MATLAB SIMULATION

MATLAB, a matrix-based mathematical program, was utilized for graphical representation of the defect detection algorithm. The needle also simulated here had a 32x32 resolution, was rotated 90 degrees and was also translated (Figure 3.1, image_y). In the second case, a defect (subtraction of a pixel) is introduced in the rotated picture (Figure A.1 image_y). In the first case all histogram differences give a zero result (see Figure 3.2, image_hist_diff and fft_hist_diff), whereas in the second case these values are different from zero (Figure A.2, image_hist_diff and fft_hist_diff). The same for cases three (Figure A.4, image_hist_diff and fft_hist_diff) and four (Figure A.6, image_hist_diff and fft_hist_diff).

A typical MATLAB session of the needle without defects rotated and translated is shown in Appendix B This example run shows the error analysis generated by MATLAB. Also shown are figures 4.1 and 4.2 showing more detailed information of the time and frequency domains. In this case all images are plotted in 3D. Complete program code and detailed run are shown in the appendix.

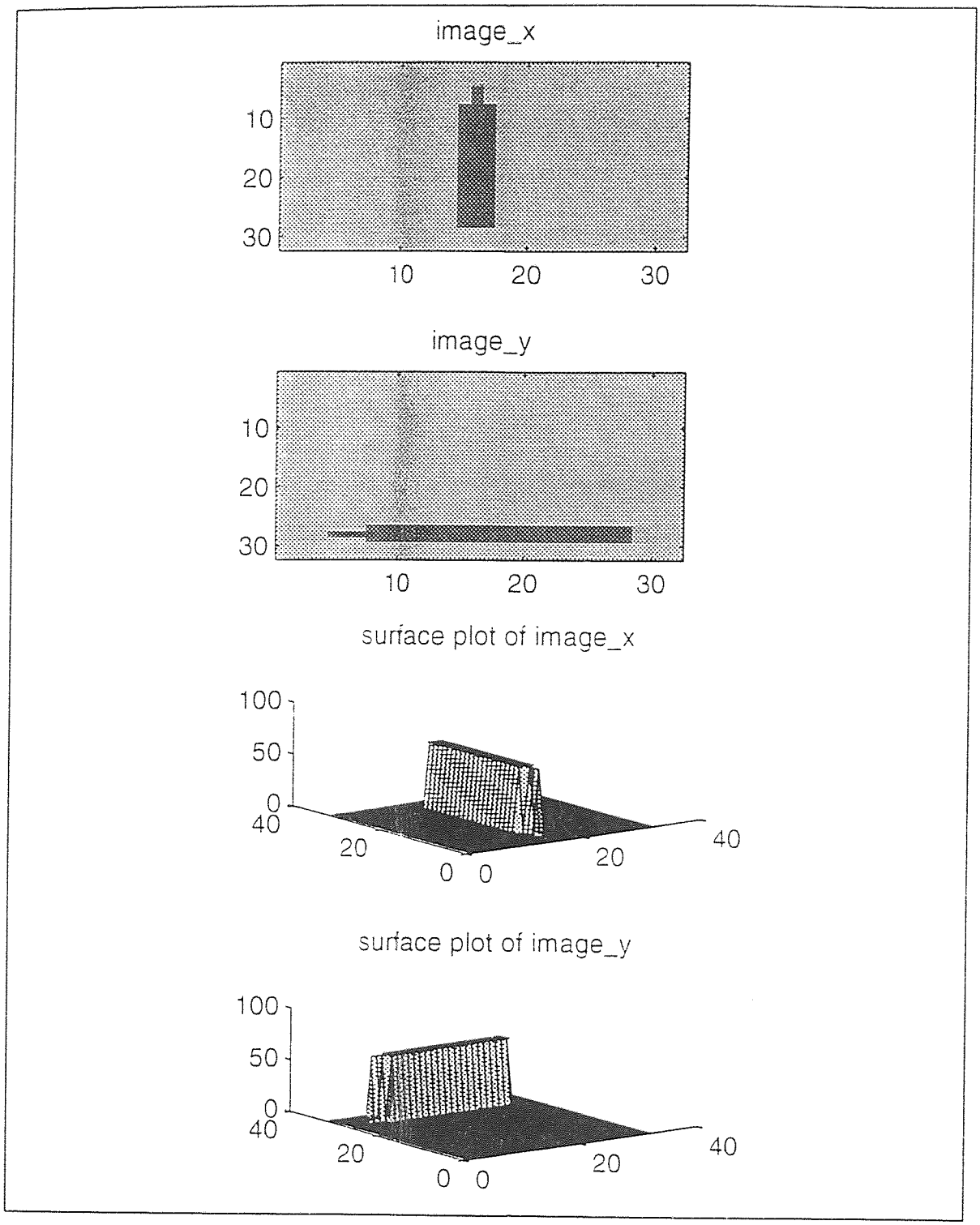

Figure 4.1 3D Intensity Plot of Needle Without Defect (time domain), Rotated and Translated

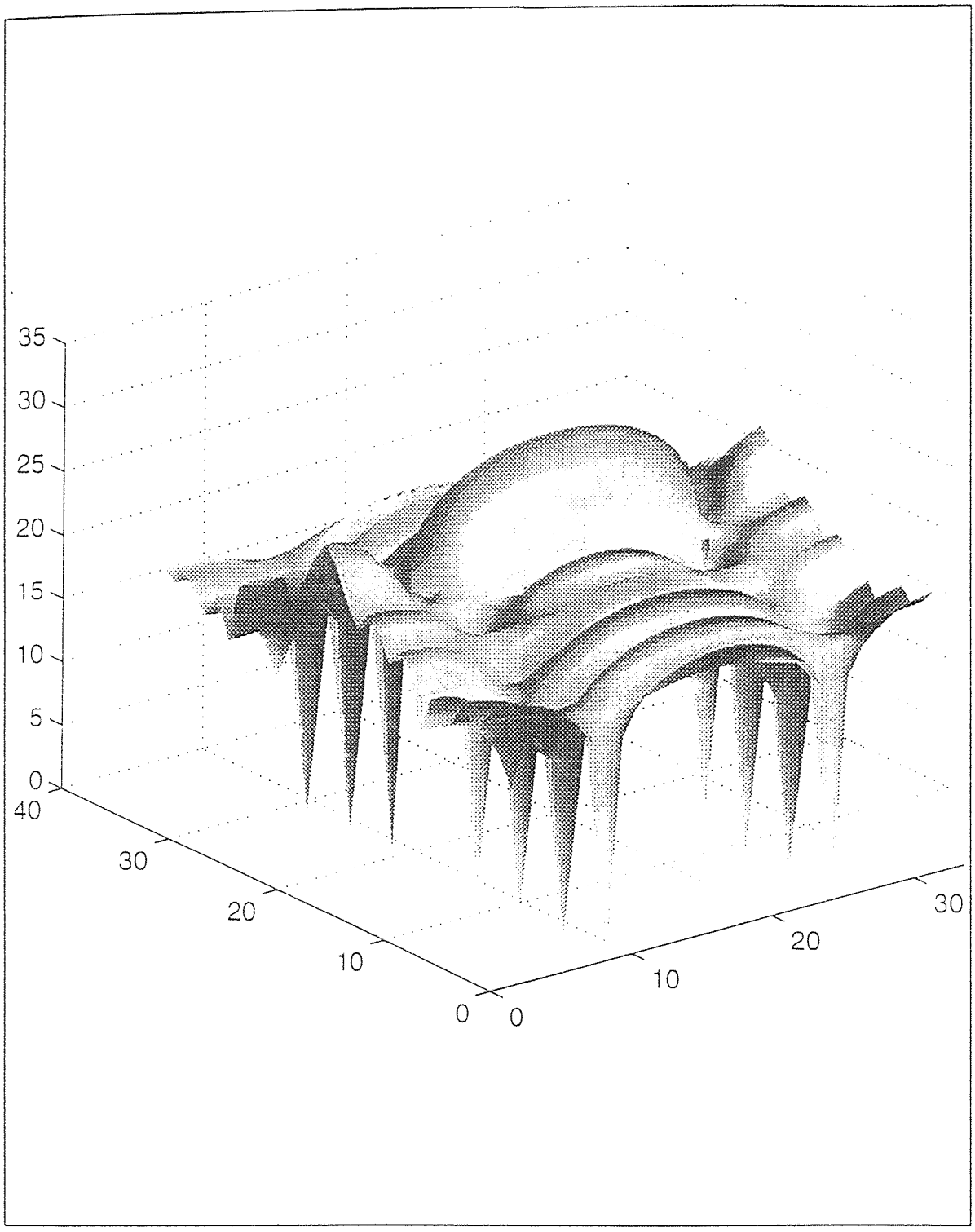

Figure 4.2 3D Intensity Plot of Needle Without Defect (frequency domain). Rotated and Translated

CHAPTER 5

PARALLEL IMPLEMENTATION AND TIMING

The algorithm is divided among 4 processors. The first two processors calculate all the time domain functions, the other two work on the frequency domain functions (figure 5.1). Interprocessor communications were used only in the FFT calculation [2]. Performance results are shown in Table 3. Impressive speedups result with parallel processing [6]. See appendix C for parallel program code and memory mapping.

Table 3 Sequential and Parallel Execution Time on TurboNet

	FFT Image A	FFT Image B	Rest of Algorithm	Total Time
One DSP	0.0375 sec	0.0375 sec	11748 usec	0.0867 sec
Four DSP	0.013 sec	0.013 sec	3953 usec	0.03 sec
Speedup	2.88	2.88	2.97	2.89

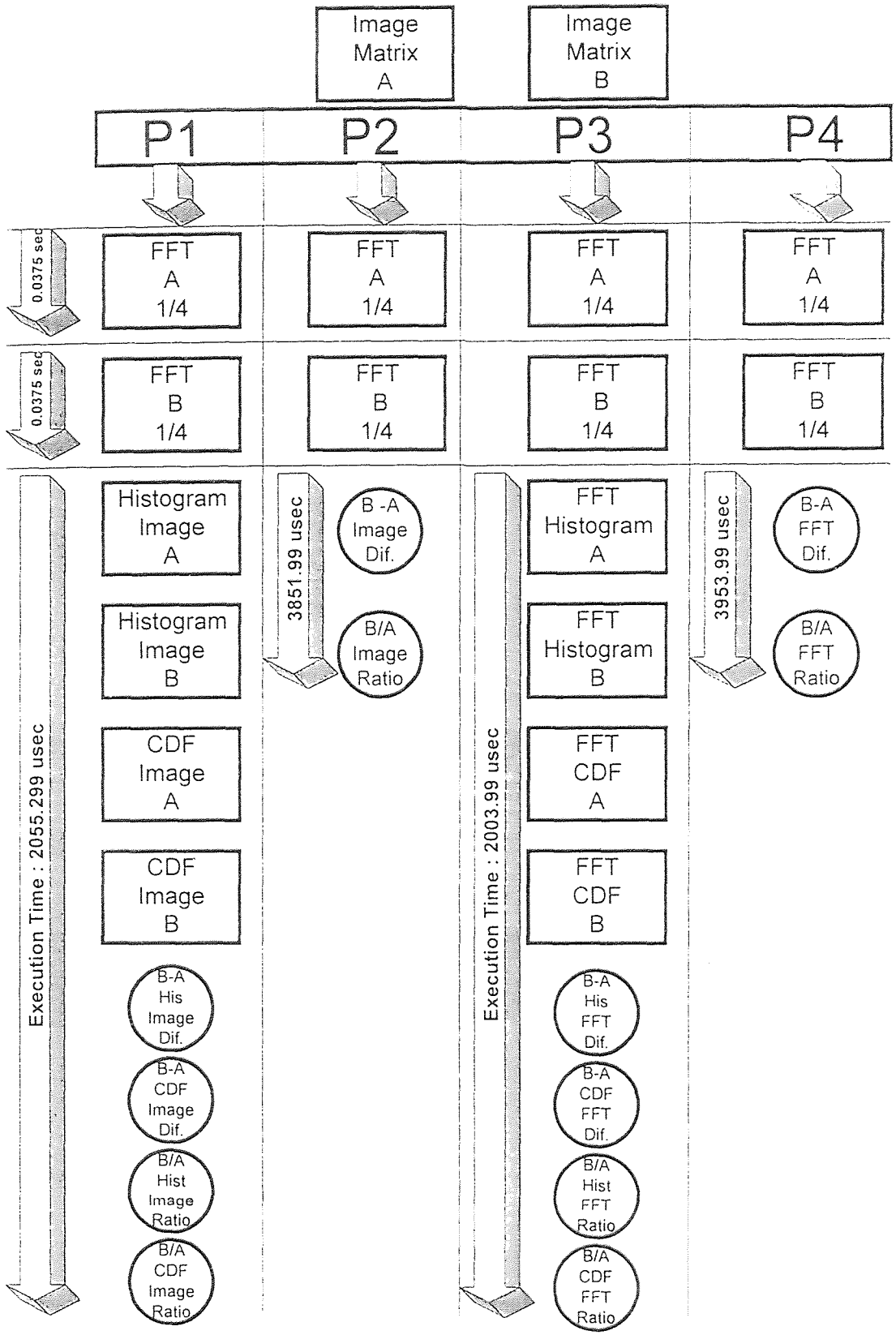

Figure 5.1 Parallel Flowchart of The Defect Algorithm

CHAPTER 6

CONCLUSION

In this thesis the performance of an algorithm for pattern recognition involving defect detection for a single object, was described and test results were presented for sequential and parallel systems. Its parallel version shows impressive speedups. The defect detection part works well in both the time and frequency domains. Both domains can be used independently or can be combined, depending on two important cases: First, two unknown objects are compared for similarity purposes. No previous knowledge and database data are available for comparison purposes and decision making. Both domains should be used and results should be compared. Second, an original object and its possible defects are kept in a database. Brute force comparison can be made by using one domain only. Depending on the application, an error factor can be set. Further study should involve pattern recognition and defect detection at any angle and at different resolutions, expansion of the algorithm, for its application in other areas [7], and defect identification.

APPENDIX A

IMAGES AND FIGURES

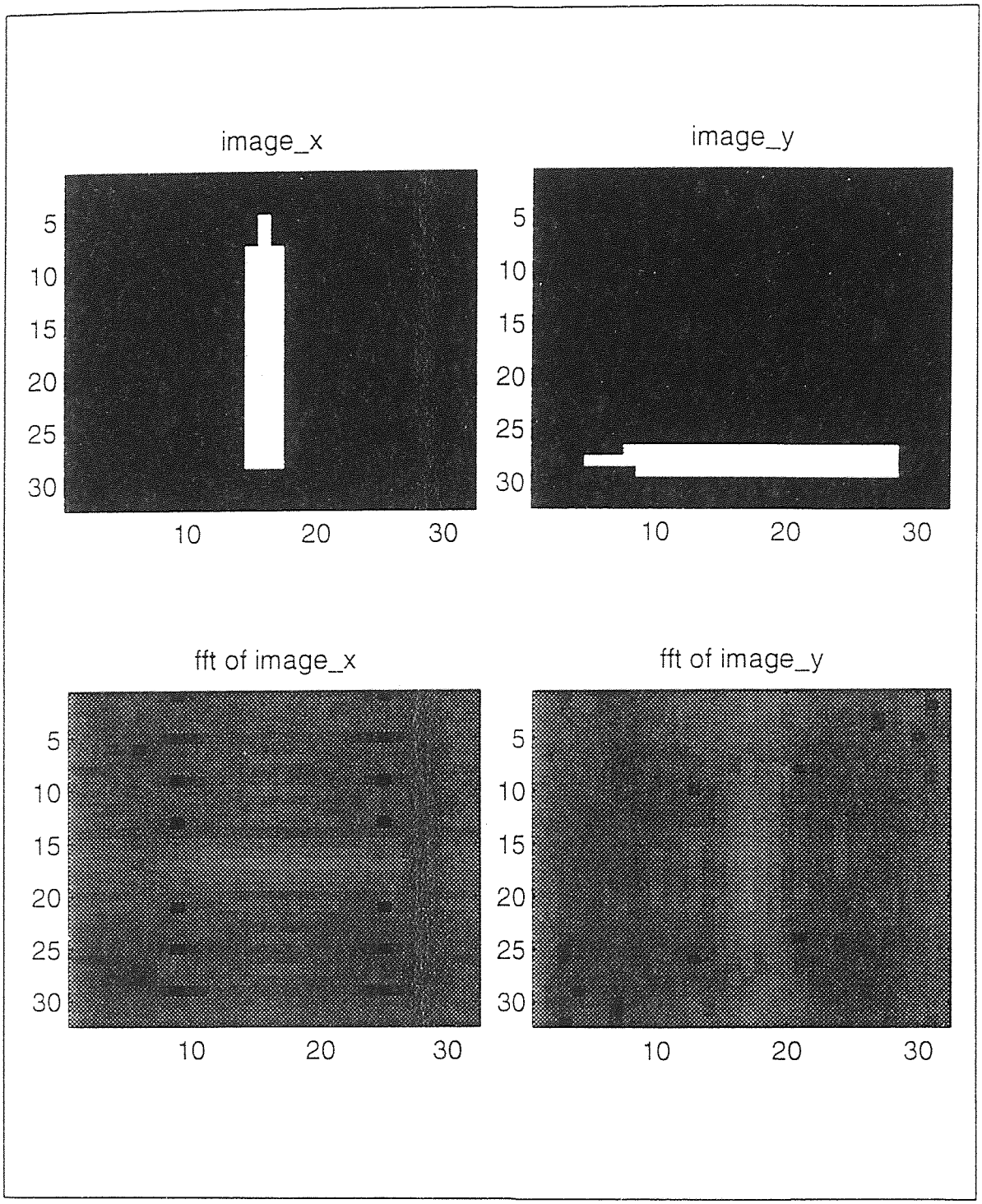

Figure A.1 Image of Needle with Defect (one pixel subtracted at the top) Rotated and Translated

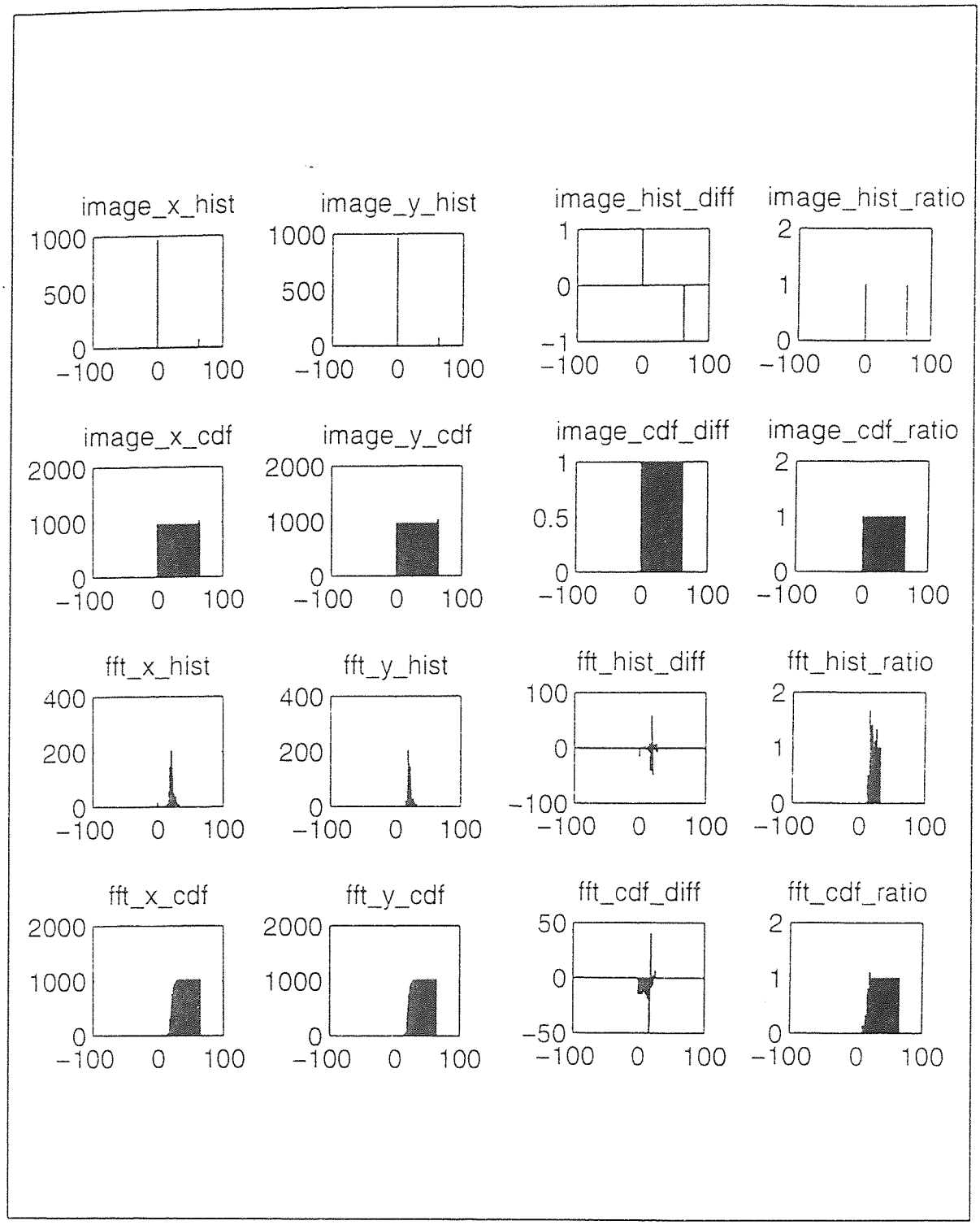

Figure A.2 Analysis of Needle with Defect (one pixel subtracted at the top) Rotated and Translated

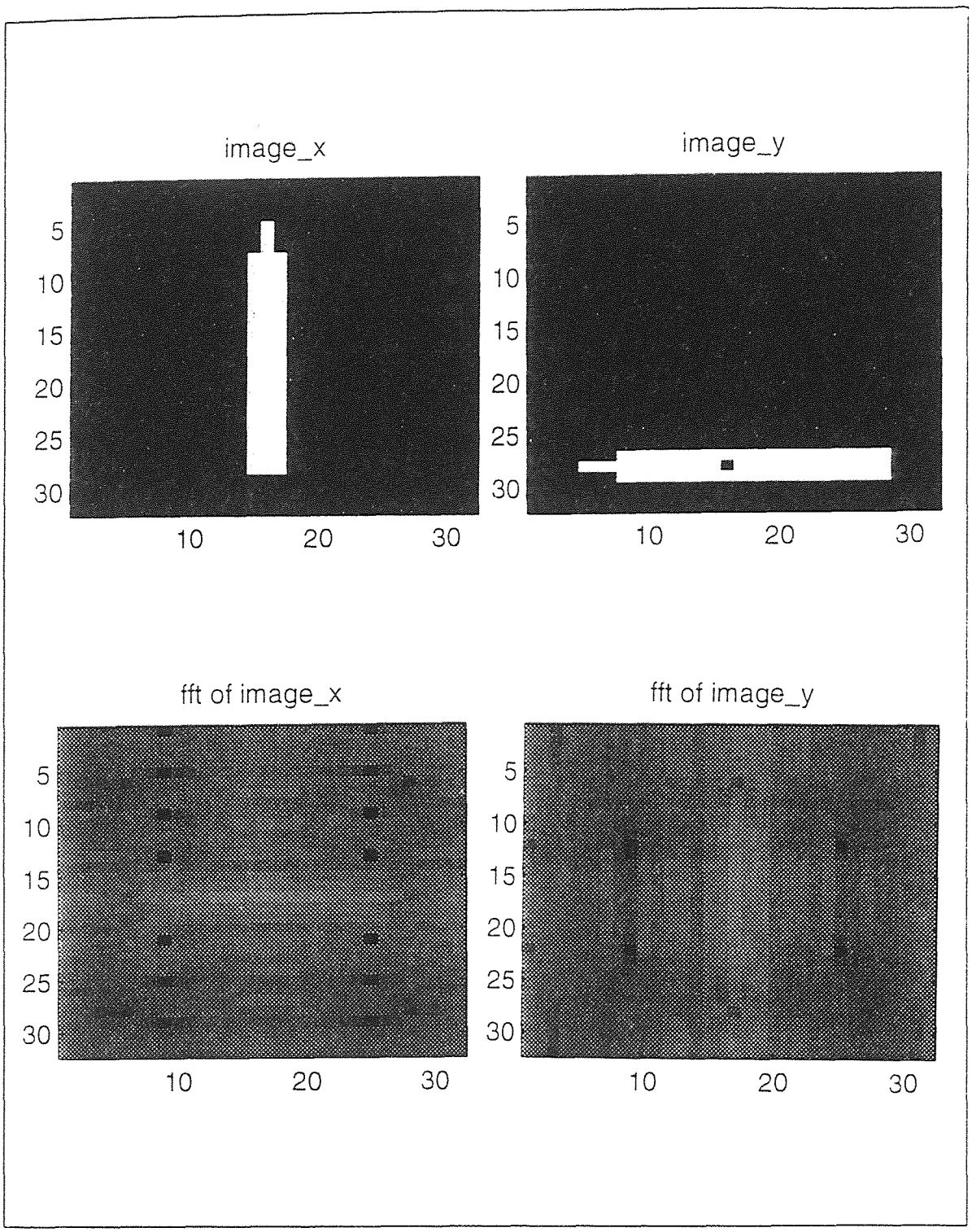

Figure A.3 Image of Needle with Defect (one pixel subtracted at the center) Rotated and Translated

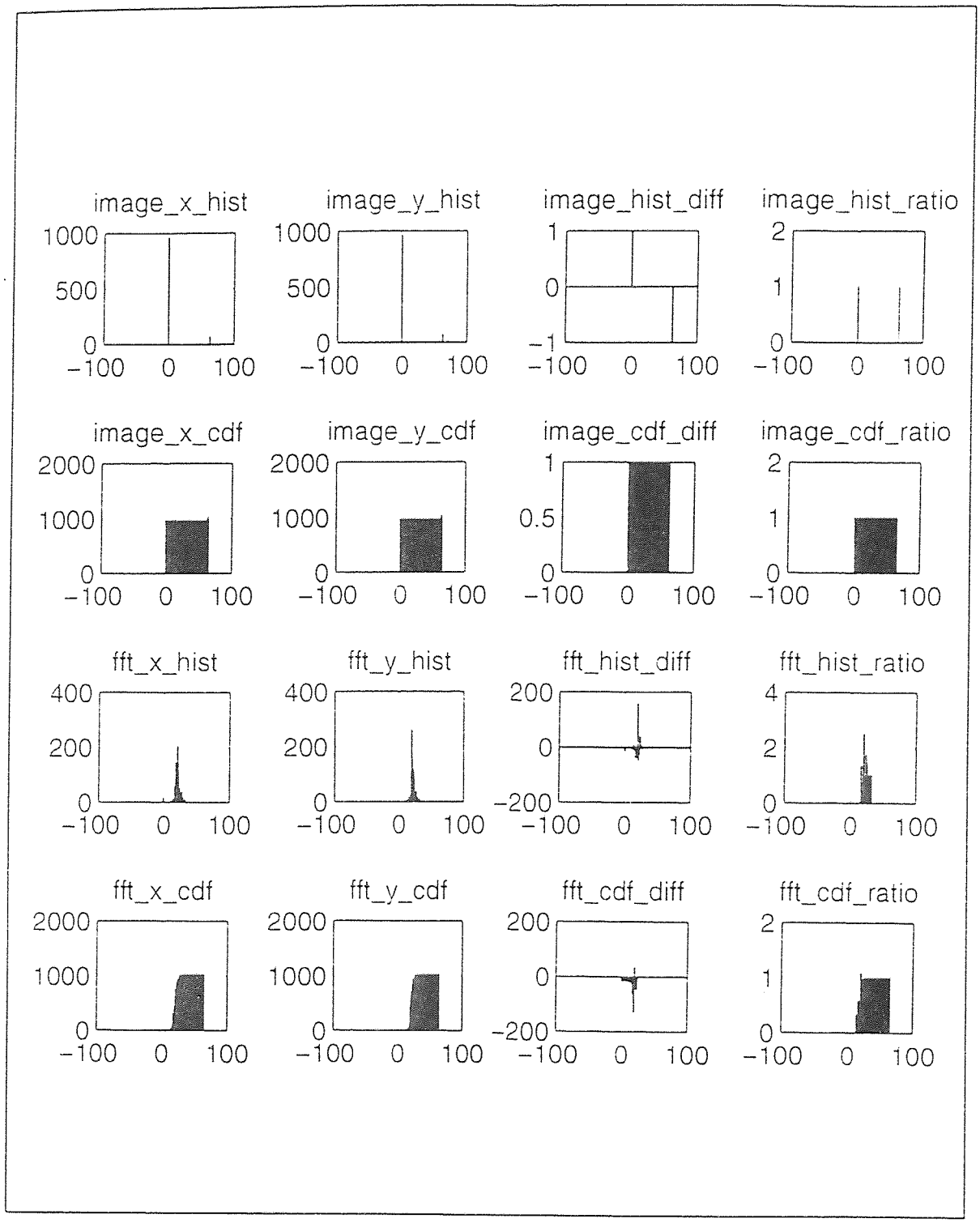

Figure A.4 Analysis of Needle with Defect (one pixel subtracted at the center) Rotated and Translated

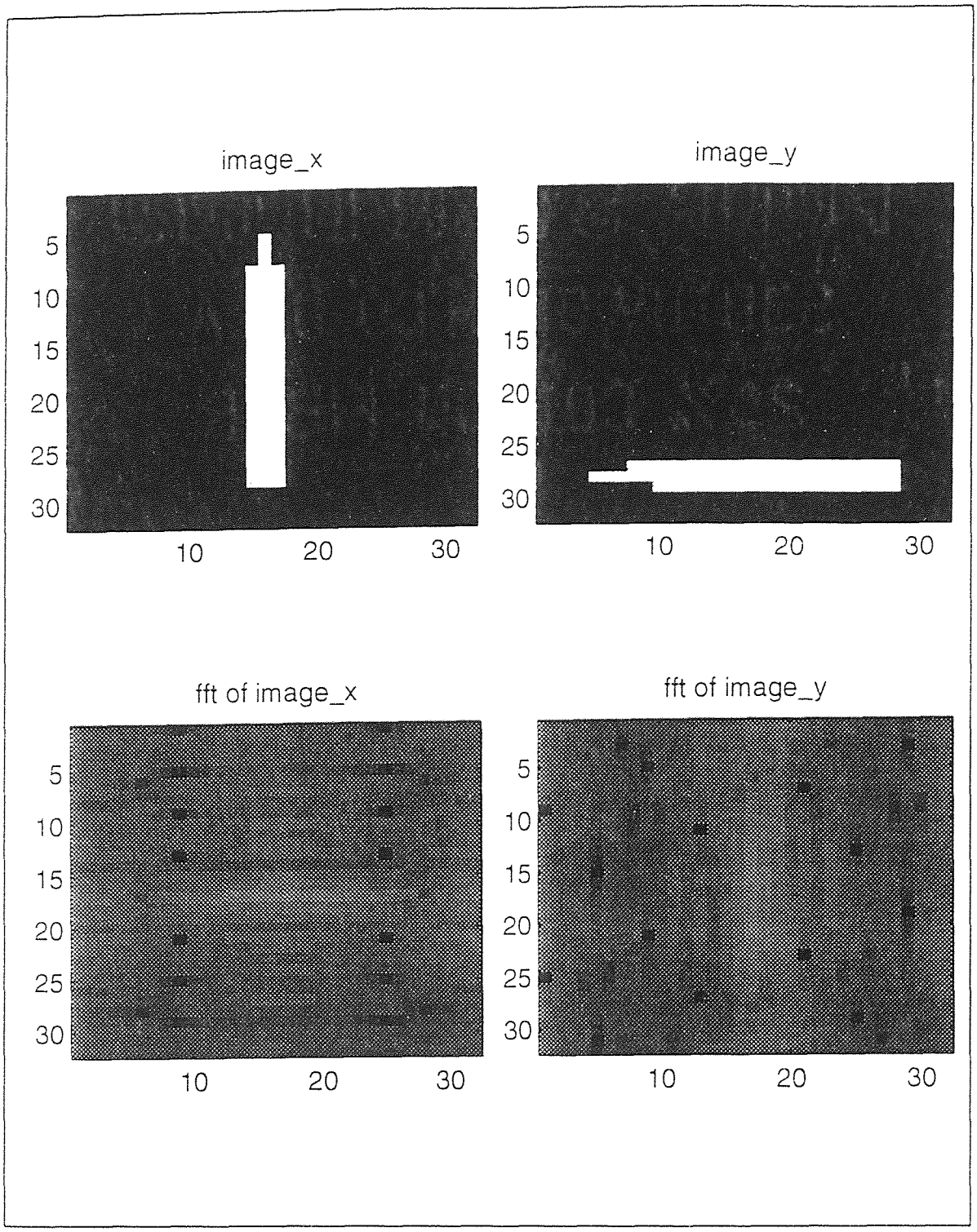

Figure A.5 Image of Needle with Defect (two pixels subtracted at the top) Rotated and Translated

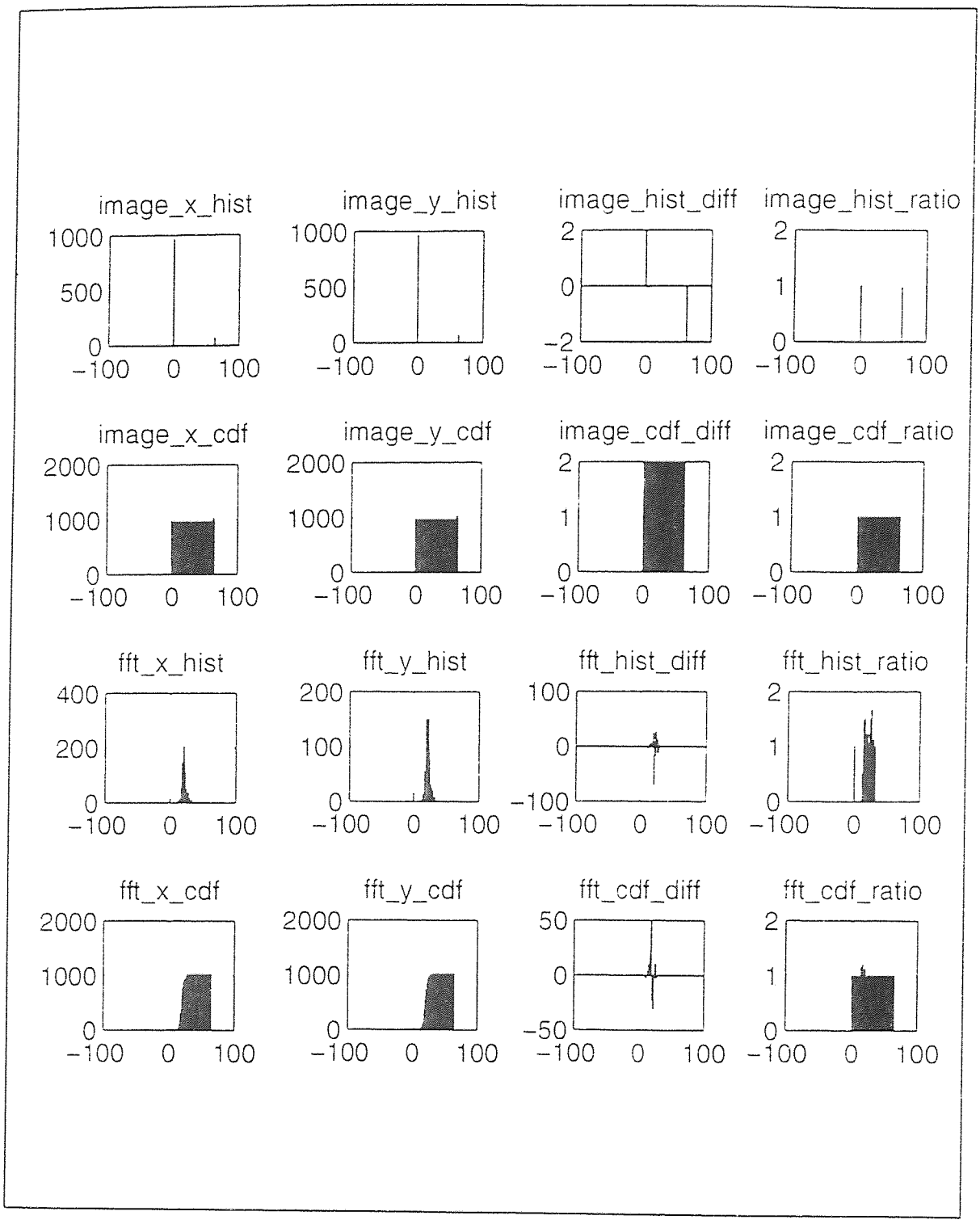

Figure A.6 Analysis of Needle with Defect (two pixels subtracted at the top) Rotated and Translated

APPENDIX B

B.1 MATLAB PROGRAM CODE

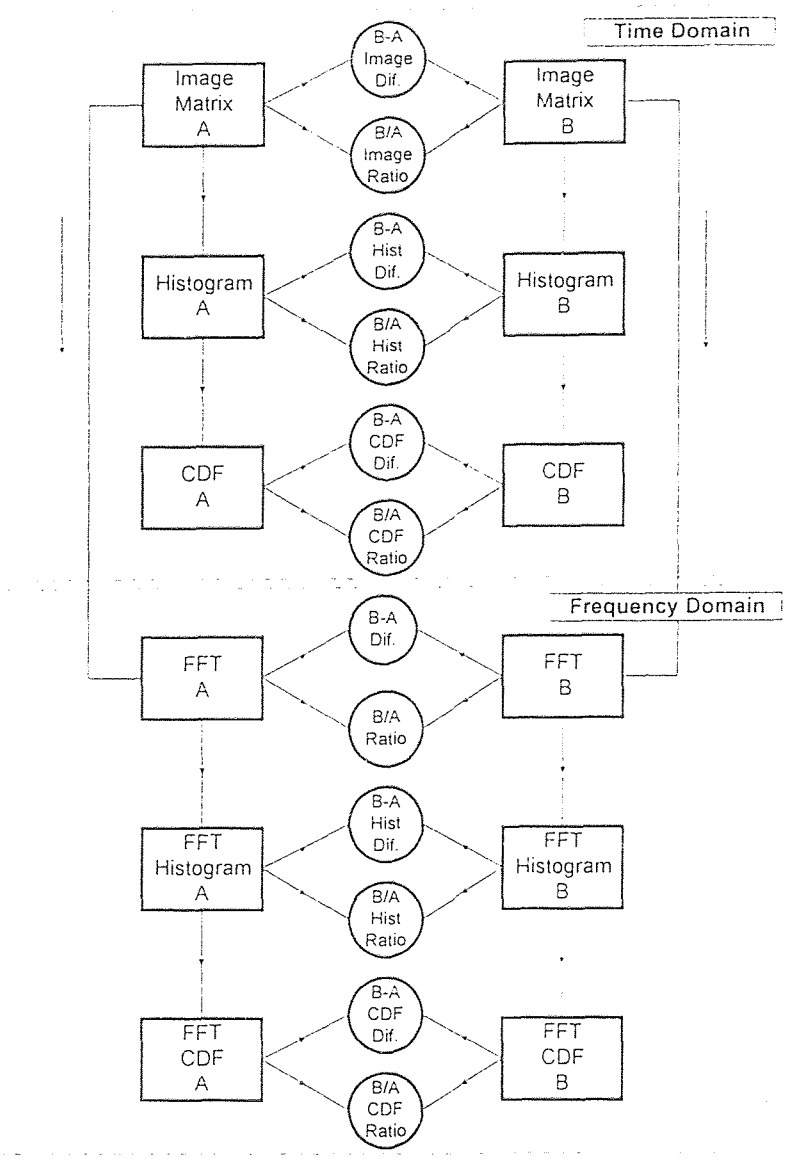

Figure B.1 Program Flowchart

Each function or block is described as an independent program. Example: FFT of A and FFT of B are two different programs. A main program named main32x32_x.m calls all the block functions into one main program. This main routine also manages all the graphics and plots. The following is a file list and its function description.

B.2 PROGRAM LISTING

This file listing follows a standard UNIX directory listing.

Table 4 File and Variable Name Coding Table
(8 File or Var. Letters +3 Letter extension)

i,f	x,y,w	-	h,c	d,r	-	e,s	x
Image FFT	Input Output Both		Histogram CDF	Difference Ratio		Error SNR	Comment File Extension

Example: *iw_hd.c*-----> histogram difference of input and output image (C code file).

```
-rw-r--r--  1 rxh5620    2239 Mar  5 03:32  cir_sec.m:      circle generator
-rw-r--r--  1 rxh5620     327 Mar  5 03:32  circ_16.m:      circle generator
-rw-r--r--  1 rxh5620     391 Mar  5 03:32  conversionx.c
-rw-r--r--  1 rxh5620     391 Mar  5 03:32  conversiony.c
-rw-r--r--  1 rxh5620     393 Mar  5 03:32  errordiff.m:    error calculation
-rw-r--r--  1 rxh5620     393 Mar  5 03:32  errorhistdiff.m
-rw-r--r--  1 rxh5620    2159 Mar  5 03:32  fft_x.m:        2D Fourier
-rw-r--r--  1 rxh5620    2212 Mar  5 03:32  fft_y.m:        2D Fourier
-rw-r--r--  1 rxh5620     677 Mar  5 03:32  fftcdfdiff.m
-rw-r--r--  1 rxh5620     686 Mar  5 03:32  fftcdfratio.m
-rw-r--r--  1 rxh5620     753 Mar  5 03:33  fftdiff.m
-rw-r--r--  1 rxh5620     704 Mar  5 03:33  ffthistdiff.m
-rw-r--r--  1 rxh5620     711 Mar  5 03:33  ffthistratio.m
-rw-r--r--  1 rxh5620     763 Mar  5 03:33  fftratio.m
-rw-r--r--  1 rxh5620     791 Mar  5 03:33  fftxabshist.m
-rw-r--r--  1 rxh5620     638 Mar  5 03:33  fftxcdf.m
-rw-r--r--  1 rxh5620     758 Mar  5 03:33  fftxhist.m
-rw-r--r--  1 rxh5620     632 Mar  5 03:33  fftycdf.m
-rw-r--r--  1 rxh5620     758 Mar  5 03:33  fftyhist.m
-rw-r--r--  1 rxh5620     657 Mar  5 03:33  fileget.m:      file transfer manager
-rw-r--r--  1 rxh5620     714 Mar  5 03:33  filesend.m      file transfer manager
-rw-r--r--  1 rxh5620    1463 Mar  5 03:33  i_ave.m
-rw-r--r--  1 rxh5620   16416 Mar  5 03:33  image1.ascii:   image data
-rw-r--r--  1 rxh5620   16416 Mar  5 03:33  image2.ascii
-rw-r--r--  1 rxh5620   16416 Mar  5 03:33  image3.ascii
-rw-r--r--  1 rxh5620     567 Mar  5 03:33  image32x32.m:          main routine
-rw-r--r--  1 rxh5620     740 Mar  5 03:33  image32x32_1.m
-rw-r--r--  1 rxh5620     741 Mar  5 03:33  image32x32_2.m
-rw-r--r--  1 rxh5620     796 Mar  5 03:33  image32x32_3.m
-rw-r--r--  1 rxh5620     794 Mar  5 03:33  image32x32_4.m
```

```
-rw-r--r--  1 rxh5620    796 Mar  5 03:33  image32x32_5.m
-rw-r--r--  1 rxh5620  16416 Mar  5 03:33  image4.ascii
-rw-r--r--  1 rxh5620  16416 Mar  5 03:33  image5.ascii
-rw-r--r--  1 rxh5620   4112 Mar  5 03:33  image6.ascii
-rw-r--r--  1 rxh5620    573 Mar  5 03:33  image_1.m:     image generator
-rw-r--r--  1 rxh5620    733 Mar  5 03:33  image_2.m
-rw-r--r--  1 rxh5620    661 Mar  5 03:33  image_3.m
-rw-r--r--  1 rxh5620    588 Mar  5 03:33  image_4.m
-rw-r--r--  1 rxh5620    652 Mar  5 03:33  image_5.m
-rw-r--r--  1 rxh5620    716 Mar  5 03:33  image_6.m
-rw-r--r--  1 rxh5620   2362 Mar  5 03:33  image_c_1.ascii
-rw-r--r--  1 rxh5620   2356 Mar  5 03:33  image_c_2.ascii
-rw-r--r--  1 rxh5620    710 Mar  5 03:33  imagecdfdiff.m
-rw-r--r--  1 rxh5620    725 Mar  5 03:33  imagecdfratio.m
-rw-r--r--  1 rxh5620    635 Mar  5 03:33  imagediff.m
-rw-r--r--  1 rxh5620    736 Mar  5 03:33  imagehistdiff.m
-rw-r--r--  1 rxh5620    750 Mar  5 03:33  imagehistratio.m
-rw-r--r--  1 rxh5620    648 Mar  5 03:33  imageratio.m
-rw-r--r--  1 rxh5620    658 Mar  5 03:33  imagexcdf.m
-rw-r--r--  1 rxh5620    769 Mar  5 03:33  imagexhist.m
-rw-r--r--  1 rxh5620    642 Mar  5 03:33  imageycdf.m
-rw-r--r--  1 rxh5620    782 Mar  5 03:33  imageyhist.m
-rw-r--r--  1 rxh5620   8648 Mar  5 03:34  main32x32_1.m
-rw-r--r--  1 rxh5620   8655 Mar  5 03:34  main32x32_1a.m
-rw-r--r--  1 rxh5620   8656 Mar  5 03:34  main32x32_2.m
-rw-r--r--  1 rxh5620   8672 Mar  5 03:34  main_rotate.m
-rw-r--r--  1 rxh5620   8581 Mar  5 03:34  main_short.m
-rw-r--r--  1 rxh5620    450 Mar  5 03:34  pfft.m
-rw-r--r--  1 rxh5620    387 Mar  5 03:34  rlogin.m
-rw-r--r--  1 rxh5620    606 Mar  5 03:34  rms_error.m
-rw-r--r--  1 rxh5620    588 Mar  5 03:34  rms_error_1d.m
-rw-r--r--  1 rxh5620   1754 Mar  5 03:34  rms_main.m
-rw-r--r--  1 rxh5620   1742 Mar  5 03:34  rotate.m:       image rotation routine
-rw-r--r--  1 rxh5620    685 Mar  5 03:34  run1.m
-rw-r--r--  1 rxh5620  56663 Mar  5 03:34  rundiff.ascii
```

B.3 PROGRAM LISTING

-rw-r--r-- 1 rxh5620 2239 Mar 5 03:32 cir_sec.m: circle generator

```
% create circular and sector matrix
% --------------------------------


.

% set variables
% -------------

for x= 1:128
for y= 1:128

image_circle(x,y) =5;
image_sector(x,y) =0:
end
end

for rings = 0:15:45


% set radius
% ----------

radius_large = 15+rings;
radius_small = 0+rings;

% set gray_level
% --------------

gray_level = rings/15+1;

% part 1
% -------
for x = - radius_large: 1: (-radius_small-1)
   nnn = floor(sqrt(radius_large^2-x^2));

for y =64 -nnn :1 : 64+ nnn
   image_circle(64+x,y) = gray_level;

end
end

% part 2
% ------

for x = - radius_small: 1: radius_small
   nnn_large = floor(sqrt(radius_large^2-x^2));
```

```
   nnn_small = floor(sqrt(radius_small^2-x^2));

for y = 64+ nnn_small :1 : 64+ nnn_large
   image_circle(64+x,y) = gray_level;
   end

for y = 64- nnn_large :1 : 64- nnn_small
   image_circle(64+x,y) = gray_level;
   end

end



% part 3
% -------
for x =  radius_small+1: 1:radius_large
   nnn = floor(sqrt(radius_large^2-x^2));

for y = 64-nnn :1 : 64+ nnn
   image_circle(64+x,y) = gray_level;

end
end



end

% count
% -----
image_circle_hist = [1:5];

for x = 1:1:5
image_circle_hist(x) =0;
end


for x = 1:128
for y = 1:128

temp = image_circle(x,y);
image_circle_hist(temp)=image_circle_hist(temp)+1;

end
end


% create image_sector
% -------------------

% set gray_level
% --------------
msize = 128
```

```
% part 1
% -------
for x  = 1: 1: msize/2

   y1 = x;
   y2 = msize/2;
   y3 = -(x-msize);
   y4 = msize ;


for y = 1:1:y1
   image_sector(x,y) = 1;
end

for y = y1+1 : 1 : y2
   image_sector(x,y) = 2;
end

for y = y2+1 : 1 : y3
   image_sector(x,y) = 3;
end

for y = y3+1 : 1 : y4
   image_sector(x,y) = 4;
end


end



% part 2
% -------
for x  = msize/2+1:1:msize

   y1 = x;
   y2 = msize/2;
   y3 = -(x-msize);
   y4 = msize ;


for y = 1:1:y3
   image_sector(x,y) = 8;
end

for y = y3+1 : 1 : y2
   image_sector(x,y) = 7;
end

for y = y2+1 : 1 : y1
   image_sector(x,y) = 6;
end
```

```
for y = y1+1 : 1 : y4
   image_sector(x,y) = 5;
end


end



% count
% -----
image_sector_hist = [1:8];

for x = 1:1:8
image_sector_hist(x) =0;
end


for x = 1:128
for y = 1:128

temp = image_sector(x,y);
image_sector_hist(temp)=image_sector_hist(temp)+1;

end
end
```

-rw-r--r-- 1 rxh5620 327 Mar 5 03:32 circ_16.m: circle generator

```
% set variables
% -------------

for x= 1:128
for y= 1:128

image_circle(x,y) =0;

end
end

% set radius
% ----------

radius = 32

% set gray_level
% --------------

gray_level = 1

for x = - radius: 1: radius
```

```
   nnn = floor(sqrt(radius^2-x^2));

for y = 64 -nnn :1 : 64 + nnn
   image_circle(64+x,y) = gray_level;

end
end
```

-rw-r--r-- 1 rxh5620 391 Mar 5 03:32 conversionx.c

```
#include <stdio.h>
#define M 16
main()
{

FILE *fp, *fx, *fopen();

int i, j;

float a[M][M], b[M][M];


fp = fopen("image1.ascii", "r");
fx = fopen("image_c_1.ascii", "w");


for (i=0; i < M; i++)
{
        for (j=0; j < M; j++)
        {
                fscanf(fp, "%f", &a[i][j]);
                printf(" %2.5f ", a[i][j]);
                fprintf(fx, " %2.5f ", a[i][j]);
        }
        printf("\n");
        fprintf(fx, "\n");
}


close(fp);
close(fx);


}
```

-rw-r--r-- 1 rxh5620 391 Mar 5 03:32 conversiony.c

```
#include <stdio.h>
#define M 16
main()
{
```

```
FILE *fp, *fx, *fopen();

int i, j;

float a[M][M],  b[M][M];


fp = fopen("image2.ascii", "r");
fx = fopen("image_c_2.ascii", "w");


for (i=0; i < M; i++)
{
        for (j=0; j < M; j++)
        {
                fscanf(fp, "%f", &a[i][j]);
                printf(" %2.5f ", a[i][j]);
                fprintf(fx, " %2.5f ", a[i][j]);
        }
        printf("\n");
        fprintf(fx, "\n");
}


close(fp);
close(fx);


}
```

-rw-r--r-- 1 rxh5620 393 Mar 5 03:32 errordiff.m: error calculation

```
% find error diff
% ---------------
echo on
%---------------------------------------
fft_hist_error - image_hist_error
%---------------------------------------
fft_hist_snr  - image_hist_snr
%---------------------------------------
fft_cdf_error  - image_cdf_error
%---------------------------------------
fft_cdf_snr   - image_cdf_snr
%---------------------------------------
echo off
```

-rw-r--r-- 1 rxh5620 393 Mar 5 03:32 errorhistdiff.m

```
% find error diff
% ---------------
echo on
```

```
%-------------------------------------
fft_hist_error - image_hist_error
%-------------------------------------
fft_hist_snr  - image_hist_snr
%-------------------------------------
fft_cdf_error  - image_cdf_error
%-------------------------------------
fft_cdf_snr   - image_cdf_snr
%-------------------------------------
echo off
```

-rw-r--r-- 1 rxh5620 2159 Mar 5 03:32 fft_x.m: 2D Fourier

```
% generate 2d fft of input image (x)
% ==================================

fftx=fft2(image_x);                          % generate fft (imaginay and real)
disp('---------------------------------------')
disp('fftx created')
disp('---------------------------------------')
disp('minimum value of fftx =')
disp(min(min(fftx)))
disp('---------------------------------------')
disp('maximum value of fftx =')
disp(max(max(fftx)))
disp('---------------------------------------')
disp(' ')


fftxabs=abs(fftx);                   % absolute value
disp('fftxabs created')
disp('---------------------------------------')
disp('minimum value of fftxabs =')
disp(min(min(fftxabs)))
disp('---------------------------------------')
disp('maximum value of fftxabs =')
disp(max(max(fftxabs)))
disp(' ')

fftxshift=fftshift(fftxabs);         % shift fft
disp('---------------------------------------')
disp('fftxshift created')
disp('---------------------------------------')
disp('minimum value of fftxshift =')
disp(min(min(fftxshift)))
disp('---------------------------------------')
disp('maximum value of fftxshift =')
disp(max(max(fftxshift)))
disp(' ')

fftxlog=1+log10(fftxshift);          % rescaled absolute value
disp('---------------------------------------')
```

```
disp('fftxlog created')
disp('-------------------------------------')
disp('minimum value of fftxlog =')
disp(min(min(fftxlog)))
disp('-------------------------------------')
disp('maximum value of fftxlog =')
disp(max(max(fftxlog)))
disp(' ')

fftxmax=max(max(fftxlog));                  % maximum value of fftlog
disp('-------------------------------------')
disp('fftxmax created')
disp(' ')

grayscalemax=64/2;                          % set to 32 for ref to y
disp(' ')

fftxscaled=(grayscalemax/fftxmax)*fftxlog; % rescaling to max gray value
% fftxscaled=fftxscaled.*image_circle:      % set circle values
disp('-------------------------------------')
disp('ffxscaled created')
disp('-------------------------------------')
disp('minimum value of fftxscaled =')
disp(min(min(fftxscaled)))
disp('-------------------------------------')
disp('maximum value of fftxscaled =')
disp(max(max(fftxscaled)))
disp('-------------------------------------')
disp('end of fft process')
```

-rw-r--r-- 1 rxh5620 2212 Mar 5 03:32 fft_y.m : 2D Fourier

```
% generate 2d fft of input image (y)
% ==================================

ffty=fft2(image_y);                         % generate fft (imaginary and real)
disp('-------------------------------------')
disp('ffty created')
disp('-------------------------------------')
disp('minimum value of ffty =')
disp(min(min(ffty)))
disp('-------------------------------------')
disp('maximum value of ffty =')
disp(max(max(ffty)))
disp(' ')


fftyabs=abs(ffty);                          % absolute value
disp('-------------------------------------')
disp('fftyabs created')
disp('-------------------------------------')
disp('minimum value of fftyabs =')
disp(min(min(fftyabs)))
```

```
disp('---------------------------------------')
disp('maximum value of fftyabs =')
disp(max(max(fftyabs)))
disp(' ')

fftyshift=fftshift(fftyabs);              % shift fft
disp('---------------------------------------')
disp('fftyshift created')
disp('---------------------------------------')
disp('minimum value of fftyshift =')
disp(min(min(fftyshift)))
disp('---------------------------------------')
disp('maximum value of fftyshift =')
disp(max(max(fftyshift)))
disp(' ')

fftylog=1+log10(fftyshift);               % rescaled absolute value
disp('---------------------------------------')
disp('fftylog created')
disp('---------------------------------------')
disp('minimum value of fftylog =')
disp(min(min(fftylog)))
disp('---------------------------------------')
disp('maximum value of fftylog =')
disp(max(max(fftylog)))
disp(' ')

fftymax=max(max(fftylog));                    % maximum value of fftlog
disp('---------------------------------------')
disp('fftymax created')
disp(' ')

grayscalemax=64/2;
disp(' ')
% note : new scale referenced to fft_x and 32 gray level scale

fftyscaled=(grayscalemax/fftxmax)*fftylog; % rescaling to max gray value
% fftyscaled=fftyscaled.*image_circle;      % set circle mode
disp('---------------------------------------')
disp('fftyscaled created')
disp('---------------------------------------')
disp('minimum value of fftyscaled =')
disp(min(min(fftyscaled)))
disp('---------------------------------------')
disp('maximum value of fftyscaled =')
disp(max(max(fftyscaled)))
disp('---------------------------------------')
disp('end of fft process')
```

-rw-r--r-- 1 rxh5620 677 Mar 5 03:32 fftcdfdiff.m

```
% fft cdf difference routine
```

```
% ----------------------------

clc

disp('----------------------------------------')
disp('calculating fft_y_cdf-fft_x_cdf')
disp('----------------------------------------')

fft_cdf_diff=fft_y_cdf-fft_x_cdf
disp('fft_cdf_diff created')
disp('----------------------------------------')
% max and min values
% ------------------

fft_cdf_diff_min=min(min(fft_cdf_diff));
disp('fft_cdf_diff_min =')
disp(fft_cdf_diff_min)
disp('----------------------------------------')


fft_cdf_diff_max=max(max(fft_cdf_diff));
disp('fft_cdf_diff_max =')
disp(fft_cdf_diff_max)
disp('----------------------------------------')
disp('end of fft_cdf_diff')
```

-rw-r--r-- 1 rxh5620 686 Mar 5 03:32 fftcdfratio.m

```
% fft cdf ratio routine
% ----------------------------

clc

disp('----------------------------------------')
disp('calculating fft_y_cdf./fft_x_cdf')
disp('----------------------------------------')

fft_cdf_ratio=fft_y_cdf./fft_x_cdf;
disp('fft_cdf_ratio created')
disp('----------------------------------------')
% max and min values
% ------------------

fft_cdf_ratio_min=min(min(fft_cdf_ratio));
disp('fft_cdf_ratio_min =')
disp(fft_cdf_ratio_min)
disp('----------------------------------------')


fft_cdf_ratio_max=max(max(fft_cdf_ratio));
disp('fft_cdf_ratio_max =')
disp(fft_cdf_ratio_max)
```

```
disp('----------------------------------------')
disp('end of fft_cdf_ratio')
```

-rw-r--r-- 1 rxh5620 753 Mar 5 03:33 fftdiff.m

```
% fft difference routine
% ----------------------

clc
% define fft i/r , abs , log or scaled
% ------------------------------------

fftx_defined=fftxscaled;
ffty_defined=fftyscaled;

disp('----------------------------------------')
disp('calculating ffty_def-fftxdef')
disp('----------------------------------------')

fft_diff=ffty_defined-fftx_defined;

disp('fft_diff created')
disp('----------------------------------------')

% max and min values
% ------------------

fft_diff_min=min(min(fft_diff));
disp('fft_diff_min =')
disp(fft_diff_min)
disp('----------------------------------------')


fft_diff_max=max(max(fft_diff));
disp('fft_diff_max =')
disp(fft_diff_max)
disp('----------------------------------------')
disp('end of fft_diff')
```

-rw-r--r-- 1 rxh5620 704 Mar 5 03:33 ffthistdiff.m

```
% fft histogram difference routine
% --------------------------------

clc

disp('----------------------------------------')
disp('calculating fft_y_hist-fft_x_hist')
disp('----------------------------------------')
```

```
fft_hist_diff=fft_y_hist-fft_x_hist;
disp('fft_hist_diff created')
disp('----------------------------------------')

% max and min values
% ------------------

fft_hist_diff_min=min(min(fft_hist_diff));
disp('fft_hist_diff_min =')
disp(fft_hist_diff_min)
disp('----------------------------------------')


fft_hist_diff_max=max(max(fft_hist_diff));
disp('fft_hist_diff_max =')
disp(fft_hist_diff_max)
disp('----------------------------------------')
disp('end of fft_hist_diff')
```

-rw-r--r-- 1 rxh5620 711 Mar 5 03:33 ffthistratio.m

```
% fft histogram ratio routine
% ---------------------------------

clc

disp('----------------------------------------')
disp('calculating fft_y_hist./fft_x_hist')
disp('----------------------------------------')

fft_hist_ratio=fft_y_hist./fft_x_hist
disp('fft_hist_ratio created')
disp('----------------------------------------')

% max and min values
% ------------------

fft_hist_ratio_min=min(min(fft_hist_ratio));
disp('fft_hist_ratio_min =')
disp(fft_hist_ratio_min)
disp('----------------------------------------')


fft_hist_ratio_max=max(max(fft_hist_ratio));
disp('fft_hist_ratio_max =')
disp(fft_hist_ratio_max)
disp('----------------------------------------')
disp('end of fft_ratio_diff')
```

-rw-r--r-- 1 rxh5620 763 Mar 5 03:33 fftratio.m

```
% fft ratio routine
% ----------------

clc
% define fft i/r , abs , log or scaled
% ------------------------------------

fftx_defined=fftxscaled;
ffty_defined=fftyscaled;

disp('----------------------------------------')
disp('calculating ffty_def/fftxdef')
disp('----------------------------------------')

fft_ratio=ffty_defined./fftx_defined;

disp('fft_ratio created')
disp('----------------------------------------')

% max and min values
% ------------------

fft_ratio_min=min(min(fft_ratio));
disp('fft_ratio_min =')
disp(fft_ratio_min)
disp('----------------------------------------')


fft_ratio_max=max(max(fft_ratio));
disp('fft_ratio_max =')
disp(fft_ratio_max)
disp('----------------------------------------')
disp('end of fft_ratio')
```

-rw-r--r-- 1 rxh5620 791 Mar 5 03:33 fftxabshist.m

```
% fft_x abs histogram
% -------------------

% msize = 16;
grayx = max(max(fftxabs))
temp = 0;


misc2=[0:grayx];
misc2=misc2';
fftx_h_axis=misc2;

for i =1:(grayx+1)
misc2(i)=0;
```

```
end

fft_x_abs_hist=misc2;

clc
disp('-------------------------------------')
disp('start histogram calculation of fft_x')
disp('-------------------------------------')

disp('matrix size  =')
disp(msize)
disp('-------------------------------------')

for row = 1:msize

for column = 1:msize

temp = fftxabs(row,column);

if (temp < 0 )
   temp = 0;
elseif ( temp > grayx )
       temp = grayx;
end

fft_x_abs_hist(temp+1) = fft_x_abs_hist(temp+1) + 1;

end

end


disp('fft_x_abs_hist created')

disp('-------------------------------------')

disp('end of histogram routine')
```

-rw-r--r-- 1 rxh5620 638 Mar 5 03:33 fftxcdf.m

```
% fft_x cdf
% ----------

% msize = 16;
grayx = 64;
temp = 0;


misc2=[0:grayx];
misc2=misc2';
fftx_h_axis=misc2;
```

```
for i =1:(grayx+1)
misc2(i)=0;
end

fft_x_cdf=misc2;

clc
disp('--------------------------------------')
disp('start histogram calculation of fft_x')
disp('--------------------------------------')

disp('matrix size  =')
disp(msize)
disp('--------------------------------------')

fft_x_cdf(1)=fft_x_hist(1);

for row = 2:(grayx+1)

fft_x_cdf(row) = fft_x_cdf(row-1) + fft_x_hist(row);


end


disp('fft_x_cdf created')

disp('--------------------------------------')

disp('end of cdf routine')
```

-rw-r--r-- 1 rxh5620 758 Mar 5 03:33 fftxhist.m

```
% fft_x histogram
% ----------------

% msize = 16;
grayx = 64;
temp = 0;


misc2=[0:grayx];
misc2=misc2';
fftx_h_axis=misc2;

for i =1:(grayx+1)
misc2(i)=0;
end

fft_x_hist=misc2;
```

```
clc
disp('-------------------------------------')
disp('start histogram calculation of fft_x')
disp('-------------------------------------')

disp('matrix size  =')
disp(msize)
disp('-------------------------------------')

for row = 1:msize

for column = 1:msize

temp = fftxscaled(row,column);

if (temp < 0 )
   temp = 0;
elseif ( temp > grayx )
       temp = grayx;
end

fft_x_hist(temp+1) = fft_x_hist(temp+1) + 1;

end

end


disp('fft_x_hist created')

disp('-------------------------------------')

disp('end of histogram routine')
```

-rw-r--r-- 1 rxh5620 632 Mar 5 03:33 fftycdf.m

```
% fft_y cdf
% ----------

% msize = 16;
grayy = 64;
temp = 0;


misc2=[0:grayy];
misc2=misc2';
ffty_h_axis=misc2;

for i =1:(grayy+1)
misc2(i)=0;
end
```

```
fft_y_cdf=misc2;

clc
disp('-------------------------------------')
disp('start cdf calculation of fft_y')
disp('-------------------------------------')

disp('matrix size  =')
disp(msize)
disp('-------------------------------------')

fft_y_cdf(1)=fft_y_hist(1);

for row = 2:(grayy+1)

fft_y_cdf(row) = fft_y_cdf(row-1) + fft_y_hist(row);


end


disp('fft_y_cdf created')

disp('-------------------------------------')

disp('end of cdf routine')
```

-rw-r--r-- 1 rxh5620 758 Mar 5 03:33 fftyhist.m

```
% fft_y histogram
% ----------------

% msize = 16;
grayx = 64;
temp = 0;


misc2=[0:grayy];
misc2=misc2';
ffty_h_axis=misc2;

for i =1:(grayy+1)
misc2(i)=0;
end

fft_y_hist=misc2;

clc
disp('-------------------------------------')
disp('start histogram calculation of fft_y')
disp('-------------------------------------')
```

```
disp('matrix size =')
disp(msize)
disp('-------------------------------------')

for row = 1:msize

for column = 1:msize

temp = fftyscaled(row,column);

if (temp < 0 )
   temp = 0;
elseif ( temp > grayy )
       temp = grayy;
end

fft_y_hist(temp+1) = fft_y_hist(temp+1) + 1;

end

end


disp('fft_y_hist created')

disp('-------------------------------------')

disp('end of histogram routine')
```

-rw-r--r-- 1 rxh5620 657 Mar 5 03:33 fileget.m: file transfer manager

```
% this routine transfers the remote ascii files generated to the local system
%
=====================================================================

clc                          % clear screen

disp('file transfer protocol to be initiated')
disp('-------------------------------------')
disp(' ')
disp('please have username and password ready')
disp(' ')
disp('connecting to remote machine')
disp('when connected get xxx.ascii from remote system')
disp('command : get xxx.ascii')
disp(' ')
disp(' ')
disp('connection established follow instructions')
disp(' ')
```

```
% unix command
!time ftp gorgona
disp(' ')
disp('connection terminated')
disp('back to MATLAB command line')
```

-rw-r--r-- 1 rxh5620 714 Mar 5 03:33 filesend.m file transfer manager

```
% this routine transfers the local ascii files generated to a remote machine
% ==========================================================================

clc                          % clear screen

disp('file transfer protocol to be initiated')
disp('-------------------------------------')
disp(' ')
disp('please have username and password ready')
disp(' ')
disp('connecting to remote machine')
disp('when connected send image1.ascii and image2.ascii to remote system')
disp('command : send image1.ascii')
disp('command : send image2.ascii')
disp(' ')
disp(' ')
disp('connection established follow instructions')
disp(' ')
% unix command
!time ftp gorgona
disp(' ')
disp('connection terminated')
disp('back to MATLAB command line')
```

-rw-r--r-- 1 rxh5620 1463 Mar 5 03:33 i_ave.m

```
% Image average circle and sector algorithm
% ------------------------------------------

% create special algorithm
% ------------------------

cir_sec

% set variables
% -------------
image_circle_sum=[1:5];
for x=1:5
image_circle_sum(x)=0;
end
```

```
image_sector_sum=[1:8];
for x=1:8
image_sector_sum(x)=0;
end



for x = 1 : 128
for y = 1 : 128


temp1 = matrix_check(x,y);
temp2 = image_circle(x,y);
temp3 = image_sector(x,y);

image_circle_sum(temp2)=image_circle_sum(temp2)+temp1;
image_sector_sum(temp3) =image_sector_sum(temp3) +temp1;

end
end
disp('-------------------------------------------------')

image_circle_hist
image_circle_sum
image_circle_average = ...
            image_circle_sum./image_circle_hist
image_sector_hist
image_sector_sum
image_sector_average = ...
            image_sector_sum./image_sector_hist


disp('-------------------------------------------------')




% replace original matrix by new average circular and sector
% ------------------------------------------------------------

% set variables
% -------------

matrix_circle_ave=rand(128);
matrix_sector_ave=rand(128);


for x = 1 : 128
for y = 1 : 128
```

```
matrix_circle_ave(x,y)=0;
matrix_sector_ave(x,y)=0;

end
end



for x = 1 : 128
for y = 1 : 128

temp1 = matrix_check(x,y);
temp2 = image_circle(x,y);
temp3 = image_sector(x,y);

matrix_circle_ave(x,y) = image_circle_average(temp2);
matrix_sector_ave(x,y) = image_sector_average(temp3);

end
end
```

-rw-r--r-- 1 rxh5620 567 Mar 5 03:33 image32x32.m: image generator/simulator

```
% Image #1
% ========

clc                         % clear screen
disp('image name..: image ')
disp('description.: vertical line')
disp('size.....: 32x32')
disp(' ')

image1=spiral(32);
image1=image1*0;            % set matrix to zero

graylevel=63;               % set maximum gray level value

for i=5:1:28
for j=16:1:16
image1(i,j)=image1(i,j)+graylevel;
end
end


disp('image1 created')

% save image as an ascii file
% ---------------------------

% save image1.ascii image1 -ascii
% unix command to display files
disp(' ')
```

```
disp(' ')
disp('image1 saved as image1.ascii ')
disp(' ')
% !cat image1.ascii
```

-rw-r--r-- 1 rxh5620 740 Mar 5 03:33 image32x32_2.m

```
% Image #2
% ========

clc                         % clear screen
disp('image name..: image2 ')
disp('description.: vertical line with defect')
disp('size.....: 32x32')
disp(' ')

image1=spiral(32);          % create 128x128 matrix
image1=image1*0;            % set matrix to zero

graylevel=63;               % set maximum gray level value
translate =0;

for i=5:1:7
for j=16+translate:1:16+translate
image1(i,j)=image1(i,j)+graylevel;
end
end

for i=8:1:28
for j=15+translate:1:17+translate
image1(i,j)=image1(i,j)+graylevel;
end
end

disp('image1 created')

% save image as an ascii file
% ---------------------------

save image1.ascii image1 -ascii
% unix command to display files
disp(' ')
!ls -l
!pwd
disp(' ')
disp('image1 saved as image1.ascii ')
disp(' ')
% !cat image1.ascii
```

-rw-r--r-- 1 rxh5620 710 Mar 5 03:33 imagecdfdiff.m

```
% image cdf difference routine
% ----------------------------

clc

disp('---------------------------------------')
disp('calculating image_y_cdf-image_x_cdf')
disp('---------------------------------------')

image_cdf_diff=image_y_cdf-image_x_cdf;
disp('image_cdf_diff created')
disp('---------------------------------------')
% max and min values
% ------------------

image_cdf_diff_min=min(min(image_cdf_diff));
disp('image_cdf_diff_min =')
disp(image_cdf_diff_min)
disp('---------------------------------------')


image_cdf_diff_max=max(max(image_cdf_diff));
disp('image_cdf_diff_max =')
disp(image_cdf_diff_max)
disp('---------------------------------------')
disp('end of image_cdf_diff')
```

-rw-r--r-- 1 rxh5620 725 Mar 5 03:33 imagecdfratio.m

```
% image cdf ratio routine
% -----------------------

clc

disp('---------------------------------------')
disp('calculating image_y_cdf./image_x_cdf')
disp('---------------------------------------')

image_cdf_ratio = image_y_cdf./image_x_cdf;
disp('image_cdf_ratio created')
disp('---------------------------------------')
% max and min values
% ------------------

image_cdf_ratio_min=min(min(image_cdf_ratio));
disp('image_cdf_ratio_min =')
disp(image_cdf_ratio_min)
disp('---------------------------------------')
```

```
image_cdf_ratio_max=max(max(image_cdf_ratio));
disp('image_cdf_ratio_max =')
disp(image_cdf_ratio_max)
disp('---------------------------------------')
disp('end of image_cdf_ratio')
```

-rw-r--r-- 1 rxh5620 635 Mar 5 03:33 imagediff.m

```
% image difference routine
% ------------------------

clc

disp('---------------------------------------')
disp('calculating image_y-image_x')
disp('---------------------------------------')

image_diff=image_y-image_x;
disp('image_diff created')
disp('---------------------------------------')

% max and min values
% ------------------

image_diff_min=min(min(image_diff));
disp('image_diff_min =')
disp(image_diff_min)
disp('---------------------------------------')


image_diff_max=max(max(image_diff));
disp('image_diff_max =')
disp(image_diff_max)
disp('---------------------------------------')
disp('end of image_diff')
```

-rw-r--r-- 1 rxh5620 736 Mar 5 03:33 imagehistdiff.m

```
% image histogram difference routine
% ---------------------------------

clc

disp('---------------------------------------')
disp('calculating image_y_hist-image_x_hist')
disp('---------------------------------------')

image_hist_diff=image_y_hist-image_x_hist;
disp('image_hist_diff created')
disp('---------------------------------------')
```

```
% max and min values
% -----------------

image_hist_diff_min=min(min(image_hist_diff));
disp('image_hist_diff_min =')
disp(image_hist_diff_min)
disp('----------------------------------------')


image_hist_diff_max=max(max(image_hist_diff));
disp('image_hist_diff_max =')
disp(image_hist_diff_max)
disp('----------------------------------------')
disp('end of image_hist_diff')
```

-rw-r--r-- 1 rxh5620 750 Mar 5 03:33 imagehistratio.m

```
% image histogram ratio routine
% ----------------------------

clc

disp('----------------------------------------')
disp('calculating image_y_hist./image_x_hist')
disp('----------------------------------------')

image_hist_ratio=image_y_hist./image_x_hist;
disp('image_hist_ratio created')
disp('----------------------------------------')

% max and min values
% -----------------

image_hist_ratio_min=min(min(image_hist_ratio));
disp('image_hist_ratio_min =')
disp(image_hist_ratio_min)
disp('----------------------------------------')


image_hist_ratio_max=max(max(image_hist_ratio));
disp('image_hist_ratio_max =')
disp(image_hist_ratio_max)
disp('----------------------------------------')
disp('end of image_hist_ratio')
```

-rw-r--r-- 1 rxh5620 648 Mar 5 03:33 imageratio.m

```
% image ratio routine
% ------------------

clc

disp('----------------------------------------')
disp('calculating image_y/image_x')
disp('----------------------------------------')

image_ratio=image_y./image_x;
disp('image_ratio created')
disp('----------------------------------------')

% max and min values
% ------------------

image_ratio_min=min(min(image_ratio));
disp('image_ratio_min =')
disp(image_ratio_min)
disp('----------------------------------------')


image_ratio_max=max(max(image_ratio));
disp('image_ratio_max =')
disp(image_ratio_max)
disp('----------------------------------------')
disp('end of image_ratio')
```

-rw-r--r-- 1 rxh5620 658 Mar 5 03:33 imagexcdf.m

```
% image_x cdf
% -----------

% msize = 16;
grayx = 64;
temp = 0;


misc2=[0:grayx];
misc2=misc2';
imagex_h_axis=misc2;

for i =1:(grayx+1)
misc2(i)=0;
end

image_x_cdf=misc2;

clc
disp('----------------------------------------')
```

```
disp('start histogram calculation of image_x')
disp('--------------------------------------')

disp('matrix size  =')
disp(msize)
disp('--------------------------------------')

image_x_cdf(1)=image_x_hist(1);

for row = 2:(grayx+1)

image_x_cdf(row) = image_x_cdf(row-1) + image_x_hist(row);


end


disp('image_x_cdf created')

disp('--------------------------------------')

disp('end of cdf routine')
```

-rw-r--r-- 1 rxh5620 769 Mar 5 03:33 imagexhist.m

```
% image_x histogram
% -----------------

% msize = 16;
grayx = 64;
temp = 0;


misc2=[0:grayx];
misc2=misc2';
imagex_h_axis=misc2;

for i =1:(grayx+1)
misc2(i)=0;
end

image_x_hist=misc2;

clc
disp('--------------------------------------')
disp('start histogram calculation of image_x')
disp('--------------------------------------')

disp('matrix size  =')
disp(msize)
disp('--------------------------------------')
```

```
for row = 1:msize

for column = 1:msize

temp = image_x(row,column);

if (temp < 0 )
   temp = 0;
elseif ( temp > grayx )
       temp = grayx;
end

image_x_hist(temp+1) = image_x_hist(temp+1) + 1;

end

end


disp('image_x_hist created')

disp('-------------------------------------')

disp('end of histogram routine')
```

-rw-r--r-- 1 rxh5620 642 Mar 5 03:33 imageycdf.m

```
% image_y cdf
% ----------

% msize = 16;
grayy = 64;
temp = 0;


misc2=[0:grayy];
misc2=misc2';
imagey_h_axis=misc2;

for i =1:(grayy+1)
misc2(i)=0;
end

image_y_cdf=misc2;

clc
disp('---------------------------------')
disp('start cdf  calculation of image_y')
disp('---------------------------------')

disp('matrix size  =')
disp(msize)
```

```
disp('--------------------------------------')

image_y_cdf(1)=image_y_hist(1)

for row = 2:(grayy+1)

image_y_cdf(row) = image_y_cdf(row-1) + image_y_hist(row);


end


disp('image_y_cdf created')

disp('--------------------------------------')

disp('end of cdf routine')
```

-rw-r--r-- 1 rxh5620 782 Mar 5 03:33 imageyhist.m

```
% image_y histogram
% ----------------

% msize = 16;
grayy = 64;
temp = 0;


misc2=[0:grayy];
misc2=misc2';
imagey_h_axis=misc2;

for i =1:(grayy+1)
misc2(i)=0;
end

image_y_hist=misc2;

clc
disp('--------------------------------------')
disp('start histogram calculation of image_y')
disp('--------------------------------------')

disp('matrix size =')
disp(msize)
disp('--------------------------------------')

for row = 1:msize

for column = 1:msize

temp = image_y(row,column);
```

```
if (temp < 0 )
   temp = 0;

elseif ( temp > grayy )
       temp = grayy;

end

image_y_hist(temp+1) = image_y_hist(temp+1) + 1;

end

end


disp('image_y_hist created')

disp('-------------------------------------')

disp('end of histogram routine')
```

-rw-r--r-- 1 rxh5620 8648 Mar 5 03:34 main32x32_1.m

```
clear

% this program runs local MATLAB routines
% which are part of the difference algorithm
% ----------------------------------------


%*************************************************************************
% set matrix size
% ---------------

msize=32
circ_16


%*************************************************************************

% create test images x and y
% --------------------------

image32x32_1            % create imagex
image_x=image1

image32x32_2            % create imagey

image_y=image2'
```

```
%*************************************************************************|
clc
figure(1)            % open figure #1
colormap(cool)       % set color parameters

% display image
% -------------


disp('---------------------------------------')
subplot(3,2,1),image(image_x)
disp('image_x created')
title('image_x')

disp('---------------------------------------')
subplot(3,2,2),surfc(image_x)
disp('surface plot of image_x created')
title('surface plot of image_x')

disp('---------------------------------------')
subplot(3,2,3),image(image_y)
disp('image_y created')
title('image_y')

disp('---------------------------------------')
subplot(3,2,4),surfc(image_y)
disp('surface plot of image_y created')
title('surface plot of image_y')

% image diff
% ----------

imagediff                % call imagediff

disp('---------------------------------------')
subplot(3,2,5),surfc(image_diff)
disp('surface plot of image_diff')
title('surfc image_diff')


% image ratio
% -----------

imageratio               % call imageratio

disp('---------------------------------------')
subplot(3,2,6),surfc(image_ratio)
disp('surface plot of image_ratio')
title('surfc image_ratio')
```

```
%*****************************************************************IV
clc
% find fft of imagex and imagey
% ----------------------------

fft_x               % 2d fft of imagex
fft_y               % 2d fft of imagey


figure(2)           % open figure #2
colormap(hsv)

% display fft
% -----------

disp('----------------------------------------')
subplot(4,2,1),image(fftxscaled)
disp('fft of image_x created')
title('fft of image_x (log scaled)')

disp('----------------------------------------')
subplot(4,2,2),surfc(fftxscaled)
disp('surfc of fftxscaled created')
title('surface plot of fftxscaled (log scaled)')

disp('----------------------------------------')
subplot(4,2,3),image(fftyscaled)
disp('fft of image_y created')
title('fft of image_y (log scaled)')

disp('----------------------------------------')
subplot(4,2,4),surfc(fftyscaled)
disp('surfc of fftyscaled created')
title('surface plot of fftyscaled (log scaled)')

% fft diff
% --------

fftdiff                 % call fftdiff


disp('----------------------------------------')
subplot(4,2,5),surfc(fft_diff)
disp('surface plot of fft_diff')
title('surfc fft_diff')


% fft ratio
% -----------

fftratio                % call fftratio

disp('----------------------------------------')
```

```
subplot(4,2,6),surfc(fft_ratio)
disp('surface plot of fft_ratio')
title('surfc fft_ratio')


% fft diff abs
% ------------

fft_abs_diff = fftyabs - fftxabs
disp('----------------------------------------')
subplot(4,2,7),surfc(fft_abs_diff)
disp('surface plot of fft_abs_diff')
title('surfc fft_abs_diff')


% fft ratio abs
% -------------

fft_abs_ratio=fftyabs./fftxabs

disp('----------------------------------------')
subplot(4,2,8),surfc(fft_abs_ratio)
disp('surface plot of fft_abs_ratio')
title('surfc fft_abs_ratio')



%*************************************************************************IV
clc
figure(3)            % open figure #3
colormap(gray)

% display images and its ffts gray level
% ---------------------------------------

colormap(gray)       % set color parameters

% display image
% -------------

disp('----------------------------------------')
subplot(2,2,1),image(image_x)
disp('image_x created')
title('image_x')


disp('----------------------------------------')
subplot(2,2,3),image(image_y)
disp('image_y created')
title('image_y')
```

```
% display fft
% -----------

disp('---------------------------------------')
subplot(2,2,2),image(fftxscaled)
disp('fft of image_x created')
title('fft of image_x')


disp('---------------------------------------')
subplot(2,2,4),image(fftyscaled)
disp('fft of image_y created')
title('fft of image_y')


%*****************************************************************||

% create image histograms
% -----------------------

imagexhist                          % call histogram routine
imageyhist

% display image histograms
% ------------------------

figure(4)                   % open window # 4

subplot(4,2,1),bar(imagex_h_axis,image_x_hist)
disp('plot of image_x_hist created')
title('image_x_hist')

disp('---------------------------------------')
subplot(4,2,2),bar(imagey_h_axis,image_y_hist)
disp('plot of image_y_hist created')
title('image_y_hist')


% create ratio and diff histograms
% --------------------------------

imagehistdiff
imagehistratio

% display diff and ratio histograms
% ---------------------------------

disp('---------------------------------------')
subplot(4,2,3),bar(imagex_h_axis,image_hist_diff)
disp('plot of image_hist_diff created')
title('image_hist_diff')

disp('---------------------------------------')
subplot(4,2,4),bar(imagex_h_axis,image_hist_ratio)
```

```
disp('plot of image_hist_ratio created')
title('image_hist_ratio')

%*****************************************************************III

% create image cdf
% ----------------

imagexcdf                       % call cdf routine
imageycdf

% display image cdf
% -----------------

figure(4)                   % open window # 4


disp('---------------------------------------')
subplot(4,2,5),bar(imagex_h_axis,image_x_cdf)
disp('plot of image_x_cdf created')
title('image_x_cdf')

disp('---------------------------------------')
subplot(4,2,6),bar(imagey_h_axis,image_y_cdf)
disp('plot of image_y_hist created')
title('image_y_cdf')


% create ratio and diff cdf
% -------------------------

imagecdfdiff
imagecdfratio

% display diff and ratio cdf
% --------------------------

disp('---------------------------------------')
subplot(4,2,7),bar(imagex_h_axis,image_cdf_diff)
disp('plot of image_cdf_diff created')
title('image_cdf_diff')

disp('---------------------------------------')
subplot(4,2,8),bar(imagex_h_axis,image_cdf_ratio)
disp('plot of image_cdf_ratio created')
title('image_cdf_ratio')

%*****************************************************************V

% create fft histograms
% ---------------------

fftxhist                    % call histogram routine
fftyhist
```

```
% display fft histograms
% ----------------------

figure(5)                    % open window #5

subplot(4,2,1),bar(fftx_h_axis,fft_x_hist)
disp('plot of fft_x_hist created')
title('fft_x_hist')

disp('----------------------------------------')
subplot(4,2,2),bar(ffty_h_axis,fft_y_hist)
disp('plot of fft_y_hist created')
title('fft_y_hist')


% create ratio and diff histograms
% --------------------------------

ffthistdiff
ffthistratio

% display diff and ratio histograms
% ---------------------------------

disp('----------------------------------------')
subplot(4,2,3),bar(fftx_h_axis,fft_hist_diff)
disp('plot of fft_hist_diff created')
title('fft_hist_diff')

disp('----------------------------------------')
subplot(4,2,4),bar(fftx_h_axis,fft_hist_ratio)
disp('plot of fft_hist_ratio created')
title('fft_hist_ratio')

%*****************************************************************VI

% create fft cdf
% --------------

fftxcdf                      % call cdf routine
fftycdf

% fft cdf plots
% -------------

figure(5)                    % open window #5


disp('----------------------------------------')
subplot(4,2,5),bar(fftx_h_axis,fft_x_cdf)
disp('plot of fft_x_cdf created')
title('fft_x_cdf')
```

```
disp('----------------------------------------')
subplot(4,2,6),bar(ffty_h_axis,fft_y_cdf)
disp('plot of fft_y_hist created')
title('fft_y_cdf')


% create ratio and diff cdf
% ------------------------

fftcdfdiff
fftcdfratio

% display diff and ratio cdf
% -------------------------

disp('----------------------------------------')
subplot(4,2,7),bar(fftx_h_axis,fft_cdf_diff)
disp('plot of fft_cdf_diff created')
title('fft_cdf_diff')

disp('----------------------------------------')
subplot(4,2,8),bar(fftx_h_axis,fft_cdf_ratio)
disp('plot of fft_cdf_ratio created')
title('fft_cdf_ratio')


%*****************************************************************
% ----------------------------
% display all variables created
% ----------------------------

clc
rms_main
errordiff
```

-rw-r--r-- 1 rxh5620 8655 Mar 5 03:34 main32x32_1a.m

```
clear

% this program runs local MATLAB routines
% which are part of the difference algorithm
% ----------------------------------------


%*****************************************************************
% set matrix size
% --------------

msize=32
circ_16
```

```
%*************************************************************************|

% create test images x and y
% -------------------------

image32x32_1              % create imagex
image_x=image1

image32x32_5              % create imagey

image_y=image5'


%*************************************************************************|
clc
figure(1)            % open figure #1
colormap(cool)       % set color parameters

% display image
% -------------


disp('---------------------------------------')
subplot(3,2,1),image(image_x)
disp('image_x created')
title('image_x')

disp('---------------------------------------')
subplot(3,2,2),surfc(image_x)
disp('surface plot of image_x created')
title('surface plot of image_x')

disp('---------------------------------------')
subplot(3,2,3),image(image_y)
disp('image_y created')
title('image_y')

disp('---------------------------------------')
subplot(3,2,4),surfc(image_y)
disp('surface plot of image_y created')
title('surface plot of image_y')

% image diff
% ----------

imagediff                 % call imagediff

disp('---------------------------------------')
subplot(3,2,5),surfc(image_diff)
disp('surface plot of image_diff')
title('surfc image_diff')
```

```
% image ratio
% -----------

imageratio                  % call imageratio

disp('----------------------------------------')
subplot(3,2,6),surfc(image_ratio)
disp('surface plot of image_ratio')
title('surfc image_ratio')


%*************************************************************************IV
clc
% find fft of imagex and imagey
% -----------------------------

fft_x               % 2d fft of imagex
fft_y               % 2d fft of imagey


figure(2)           % open figure #2
colormap(hsv)

% display fft
% -----------

disp('----------------------------------------')
subplot(4,2,1),image(fftxscaled)
disp('fft of image_x created')
title('fft of image_x (log scaled)')

disp('----------------------------------------')
subplot(4,2,2),surfc(fftxscaled)
disp('surfc of fftxscaled created')
title('surface plot of fftxscaled (log scaled)')

disp('----------------------------------------')
subplot(4,2,3),image(fftyscaled)
disp('fft of image_y created')
title('fft of image_y (log scaled)')

disp('----------------------------------------')
subplot(4,2,4),surfc(fftyscaled)
disp('surfc of fftyscaled created')
title('surface plot of fftyscaled (log scaled)')

% fft diff
% --------

fftdiff                 % call fftdiff
```

```
disp('----------------------------------------')
subplot(4,2,5),surfc(fft_diff)
disp('surface plot of fft_diff')
title('surfc fft_diff')


% fft ratio
% -----------

fftratio                    % call fftratio

disp('----------------------------------------')
subplot(4,2,6),surfc(fft_ratio)
disp('surface plot of fft_ratio')
title('surfc fft_ratio')


% fft diff abs
% ------------

fft_abs_diff = fftyabs - fftxabs
disp('----------------------------------------')
subplot(4,2,7),surfc(fft_abs_diff)
disp('surface plot of fft_abs_diff')
title('surfc fft_abs_diff')


% fft ratio abs
% -------------

fft_abs_ratio=fftyabs./fftxabs

disp('----------------------------------------')
subplot(4,2,8),surfc(fft_abs_ratio)
disp('surface plot of fft_abs_ratio')
title('surfc fft_abs_ratio')



%*****************************************************************IV
clc
figure(3)           % open figure #3
colormap(gray)

% display images and its ffts gray level
% ---------------------------------------

colormap(gray)      % set color parameters

% display image
% -------------

disp('----------------------------------------')
subplot(2,2,1),image(image_x)
```

```
disp('image_x created')
title('image_x')


disp('----------------------------------------')
subplot(2,2,2),image(image_y)
disp('image_y created')
title('image_y')



% display fft
% ------4----

disp('----------------------------------------')
subplot(2,2,3),image(fftxscaled)
disp('fft of image_x created')
title('fft of image_x')


disp('----------------------------------------')
subplot(2,2,4),image(fftyscaled)
disp('fft of image_y created')
title('fft of image_y')


%*****************************************************************II

% create image histograms
% -----------------------

imagexhist                          % call histogram routine
imageyhist

% display image histograms
% ------------------------

figure(4)                   % open window # 4

subplot(4,4,1),bar(imagex_h_axis,image_x_hist)
disp('plot of image_x_hist created')
title('image_x_hist')

disp('----------------------------------------')
subplot(4,4,2),bar(imagey_h_axis,image_y_hist)
disp('plot of image_y_hist created')
title('image_y_hist')


% create ratio and diff histograms
% --------------------------------

imagehistdiff
```

```
imagehistratio

% display diff and ratio histograms
% --------------------------------

disp('----------------------------------------')
subplot(4,4,3),bar(imagex_h_axis,image_hist_diff)
disp('plot of image_hist_diff created')
title('image_hist_diff')

disp('----------------------------------------')
subplot(4,4,4),bar(imagex_h_axis,image_hist_ratio)
disp('plot of image_hist_ratio created')
title('image_hist_ratio')

%*****************************************************************!!!

% create image cdf
% ----------------

imagexcdf                       % call cdf routine
imageycdf

% display image cdf
% -----------------

%figure(4)                      % open window # 4


disp('----------------------------------------')
subplot(4,4,5),bar(imagex_h_axis,image_x_cdf)
disp('plot of image_x_cdf created')
title('image_x_cdf')

disp('----------------------------------------')
subplot(4,4,6),bar(imagey_h_axis,image_y_cdf)
disp('plot of image_y_cdf created')
title('image_y_cdf')


% create ratio and diff cdf
% -------------------------

imagecdfdiff
imagecdfratio

% display diff and ratio cdf
% --------------------------
disp('----------------------------------------')
subplot(4,4,7),bar(imagex_h_axis,image_cdf_diff)
disp('plot of image_cdf_diff created')
title('image_cdf_diff')

disp('----------------------------------------')
```

```
subplot(4,4,8),bar(imagex_h_axis,image_cdf_ratio)
disp('plot of image_cdf_ratio created')
title('image_cdf_ratio')

%*****************************************************************V

% create fft histograms
% ---------------------

fftxhist                    % call histogram routine
fftyhist

% display fft histograms
% ----------------------

%figure(5)                          % open window #5

subplot(4,4,9),bar(fftx_h_axis,fft_x_hist)
disp('plot of fft_x_hist created')
title('fft_x_hist')

disp('---------------------------------------')
subplot(4,4,10),bar(ffty_h_axis,fft_y_hist)
disp('plot of fft_y_hist created')
title('fft_y_hist')


% create ratio and diff histograms
% --------------------------------

ffthistdiff
ffthistratio

% display diff and ratio histograms
% ---------------------------------

disp('---------------------------------------')
subplot(4,4,11),bar(fftx_h_axis,fft_hist_diff)
disp('plot of fft_hist_diff created')
title('fft_hist_diff')

disp('---------------------------------------')
subplot(4,4,12),bar(fftx_h_axis,fft_hist_ratio)
disp('plot of fft_hist_ratio created')
title('fft_hist_ratio')

%*****************************************************************VI

% create fft cdf
% --------------

fftxcdf                     % call cdf routine
fftycdf
```

```
% fft cdf plots
% -----------------

%figure(5)                          % open window #5


disp('---------------------------------------')
subplot(4,4,13),bar(fftx_h_axis,fft_x_cdf)
disp('plot of fft_x_cdf created')
title('fft_x_cdf')

disp('---------------------------------------')
subplot(4,4,14),bar(ffty_h_axis,fft_y_cdf)
disp('plot of fft_y_hist created')
title('fft_y_cdf')


% create ratio and diff cdf
% ------------------------

fftcdfdiff
fftcdfratio

% display diff and ratio cdf
% -------------------------

disp('---------------------------------------')
subplot(4,4,15),bar(fftx_h_axis,fft_cdf_diff)
disp('plot of fft_cdf_diff created')
title('fft_cdf_diff')

disp('---------------------------------------')
subplot(4,4,16),bar(fftx_h_axis,fft_cdf_ratio)
disp('plot of fft_cdf_ratio created')
title('fft_cdf_ratio')


%*****************************************************************
% ----------------------------
% display all variables created
% ----------------------------

clc
rms_main
errordiff
```

-rw-r--r-- 1 rxh5620 606 Mar 5 03:34 rms_error.m

```
% calculate rms and error
% ----------------------

% msize = 16;                       % matrix size
m=msize;
```

```
n=msize;
error_sum = 0;
error_sum_squared = 0;
error = output_image - input_image;
error_squared=error.^2;
out_image_sq = (output_image).^2;
out_image_ss = 0;


for x = 1 : (msize)
for y = 1 : (msize)

out_image_ss = out_image_sq(x,y) + out_image_ss;
error_sum = error(x,y) + error_sum ;
error_sum_squared = error_squared(x,y) + error_sum_squared;

end
end

% disp('----------------------------------------')
error_rms = ((1/(msize*msize))*error_sum_squared)^(1/2);

snr_rms = (out_image_ss)/(error_sum_squared);
```

-rw-r--r-- 1 rxh5620 588 Mar 5 03:34 rms_error_1d.m

```
% calculate rms and error 1d
% -------------------------

gray_error= 64;                    % matrix size
m=gray_error;
n=1;
error_sum = 0;
error_sum_squared = 0;
error = output_image - input_image;
error_squared=error.^2;
out_image_sq = (output_image).^2;
out_image_ss = 0;

for x = 1 : (gray_error)

out_image_ss = out_image_sq(x) + out_image_ss;
error_sum = error(x) + error_sum ;
error_sum_squared = error_squared(x) + error_sum_squared;

end
end

% disp('----------------------------------------')
error_rms = ((1/(msize))*error_sum_squared)^(1/2);

snr_rms = (out_image_ss)/(error_sum_squared);
```

-rw-r--r-- 1 rxh5620 1754 Mar 5 03:34 rms_main.m

```
% display numerically errors
% -------------------------

disp('----------------------------------------')
disp('Image Error')
disp('===========')
output_image = image_y;
input_image  = image_x;
rms_error
image_error = error_rms
image_snr   = snr_rms


disp('----------------------------------------')
disp('Image Histogram Error')
disp('======================')
output_image = image_y_hist;
input_image  = image_x_hist;
rms_error_1d
image_hist_error = error_rms
image_hist_snr   = snr_rms

disp('----------------------------------------')
disp('Image CDF Error')
disp('===============')
output_image = image_y_cdf;
input_image  = image_x_cdf;
rms_error_1d
image_cdf_error = error_rms
image_cdf_snr   = snr_rms

disp('----------------------------------------')
disp('FFT Error')
disp('=========')
output_image = fftyabs;
input_image  = fftxabs;
rms_error
fftabs_error = error_rms
fftabs_snr   = snr_rms

% eliminate inf
for x= 1:msize
for y= 1:msize
if (fftyscaled(x,y)<0)
   fftyscaled(x,y) = 0;
   elseif (fftyscaled(x,y)>64)
         fftyscaled(x,y)=64;
end

if (fftxscaled(x,y)<0)
   fftxscaled(x,y) = 0;
```

```
   elseif (fftxscaled(x,y)>64)
         fftxscaled(x,y)=64;
end

end
end

output_image = fftyscaled;
input_image  = fftxscaled;
rms_error
fftscaled_error = error_rms
fftscaled_snr   = snr_rms

disp('----------------------------------------')
disp('FFT Histogram Error')
disp('====================')
output_image = fft_y_hist;
input_image  = fft_x_hist;
rms_error_1d
fft_hist_error = error_rms
fft_hist_snr   = snr_rms

disp('----------------------------------------')
disp('FFT CDF Error')
disp('=============')
output_image = fft_y_cdf;
input_image  = fft_x_cdf;
rms_error_1d
fft_cdf_error = error_rms
fft_cdf_snr   = snr_rms
```

-rw-r--r-- 1 rxh5620 1742 Mar 5 03:34 rotate.m: image rotation routine

```
% 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
% - - - - - - - - - - - - - - - -



% plot
% ----

image_1
a=image1;
f=1
figure(f)
colormap(gray)
subplot(5,5,1),image(a)



for times=1:8
```

```
%
display=times


b= a*0;
matrix_size=128 ;
rotation = -(pi*times/8);



for i= 1:matrix_size
for j= 1:matrix_size


    y_original = i;
    x_original = j;

% create rotational equations

% 1.translate to 0,0 origin
% -------------------------

x_translate = x_original - (matrix_size/2);
y_translate = y_original - (matrix_size/2);

% 2.find polar coordinates
% ------------------------

r = sqrt( (x_translate)^2  + (y_translate)^2 );

if x_translate == 0
angle = atan(y_translate/10e-100);
else
angle = atan(y_translate/x_translate);
end

% 3. set rotation
% ---------------

x_rotated = r*cos(angle+rotation)+(matrix_size/2);
y_rotated = r*sin(angle+rotation)+(matrix_size/2);



if a(i,j) > 0

   b(y_rotated,x_rotated)=a(i,j);

    end
```

```
  end

end



% plot
% ----

.
if times == 1
  save rotpi1_8 b /ascii
elseif times == 2
  save rotpi2_8 b /ascii
elseif times == 3
  save rotpi3_8 b /ascii
elseif times == 4
  save rotpi4_8 b /ascii
elseif times == 5
  save rotpi5_8 b /ascii
elseif times == 6
  save rotpi6_8 b /ascii
elseif times == 7
  save rotpi7_8 b /ascii
elseif times == 8
  save rotpi8_8 b /ascii
elseif times == 9
  save rotpi9_8 b /ascii
elseif times ==10
  save rotpi10_8 b /ascii
elseif times ==11
  save rotpi11_8 b /ascii
elseif times ==12
  save rotpi12_8 b /ascii
elseif times ==13
  save rotpi13_8 b /ascii
elseif times ==14
  save rotpi14_8 b /ascii
elseif times ==15
  save rotpi15_8 b /ascii
elseif times ==16
  save rotpi16_8 b /ascii
end

figure(f)
colormap(gray)
subplot(5,5,times+1),image(b)

 end
```

B.3 EXAMPLE RUN

```
Script started on Sun Mar  5 01:38:38 1995
fugue% matlab4.2

                    < M A T L A B (R) >
          (c) Copyright 1984-94 The MathWorks, Inc.
                  All Rights Reserved
                     Version 4.2
                     Mar 29 1994

Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe

>> main32x32_1a

msize =

   32


radius =

   32


gray_level =

    1

image name..: image2
description.: vertical line with defect
size.....: 32x32

image1 created

total 1730
-rwxrwxrwx  1 rxh5620    24576 Nov 21 17:17 a.out
-rw-rw-rw-  1 rxh5620       76 Nov 21 17:17 allmain.m
-rw-rw-rw-  1 rxh5620      260 Nov 21 17:17 b.ascii
-rw-r--r--  1 rxh5620     2239 Dec  5 01:11 cir_sec.m
-rw-r--r--  1 rxh5620      327 Dec  1 16:16 circ_16.m
-rw-rw-rw-  1 rxh5620       39 Nov 21 17:17 control.m
-rw-rw-rw-  1 rxh5620      391 Nov 21 17:17 conversionx.c
-rwxrwxrwx  1 rxh5620    24576 Nov 21 17:17 conversionx.out
-rw-rw-rw-  1 rxh5620      391 Nov 21 17:17 conversiony.c
-rwxrwxrwx  1 rxh5620    24576 Nov 21 17:17 conversiony.out
-rw-rw-rw-  1 rxh5620     1744 Nov 21 17:17 data16x16.m
-rw-rw-rw-  1 rxh5620      575 Nov 21 17:17 data4x4.m
-rw-rw-rw-  1 rxh5620     3059 Nov 21 17:17 data4x4.rec
-rw-rw-rw-  1 rxh5620      393 Nov 21 17:17 errordiff.m
```

```
-rw-rw-rw-  1 rxh5620     393 Nov 21 17:17 errorhistdiff.m
-rw-rw-rw-  1 rxh5620    2159 Dec  5 01:43 fft_x.m
-rw-rw-rw-  1 rxh5620    2212 Dec  5 01:43 fft_y.m
-rw-rw-rw-  1 rxh5620     677 Nov 21 17:17 fftcdfdiff.m
-rw-rw-rw-  1 rxh5620     686 Nov 30 16:15 fftcdfratio.m
-rw-rw-rw-  1 rxh5620     753 Nov 30 13:30 fftdiff.m
-rw-rw-rw-  1 rxh5620     704 Nov 30 16:14 ffthistdiff.m
-rw-rw-rw-  1 rxh5620     711 Nov 21 17:17 ffthistratio.m
-rw-rw-rw-  1 rxh5620     763 Nov 30 13:32 fftratio.m
-rw-rw-rw-  1 rxh5620     791 Nov 28 22:28 fftxabshist.m
-rw-rw-rw-  1 rxh5620     638 Nov 28 22:30 fftxcdf.m
-rw-rw-rw-  1 rxh5620     758 Nov 28 22:35 fftxhist.m
-rw-rw-rw-  1 rxh5620     632 Nov 28 22:35 fftycdf.m
-rw-rw-rw-  1 rxh5620     758 Nov 28 22:36 fftyhist.m
-rw-rw-rw-  1 rxh5620  188062 Feb 25 13:11 fig1.ps
-rw-rw-rw-  1 rxh5620  432784 Feb 25 13:12 fig2.ps
-rw-rw-rw-  1 rxh5620   16630 Feb 25 13:15 fig3.ps
-rw-r--r--  1 rxh5620   16630 Mar  3 00:29 fig3_bone.ps
-rw-rw-rw-  1 rxh5620   16630 Mar  3 02:47 fig3_bright.ps
-rw-r--r--  1 rxh5620   16596 Mar  3 00:37 fig3_eps.eps
-rw-r--r--  1 rxh5620    8530 Mar  3 01:45 fig3_gif.gif
-rw-r--r--  1 rxh5620   16630 Mar  3 00:18 fig3_hot.ps
-rw-rw-rw-  1 rxh5620   16630 Feb 25 13:25 fig3_original.ps
-rw-r--r--  1 rxh5620   42236 Mar  3 01:31 fig3_pcx.pcx
-rw-rw-rw-  1 rxh5620   16630 Feb 28 02:36 fig3_prism.ps
-rw-r--r--  1 rxh5620   16630 Mar  3 03:07 fig3_thesis.ps
-rw-rw-rw-  1 rxh5620   16630 Feb 25 13:37 fig3a.ps
-rw-r--r--  1 rxh5620   16630 Mar  3 03:13 fig3a_thesis.ps
-rw-r--r--  1 rxh5620   16630 Mar  3 03:20 fig3b_thesis.ps
-rw-r--r--  1 rxh5620   16630 Mar  3 03:28 fig3c_thesis.ps
-rw-rw-rw-  1 rxh5620   35602 Feb 25 13:16 fig4.ps
-rw-r--r--  1 rxh5620   34439 Mar  3 03:07 fig4_thesis.ps
-rw-rw-rw-  1 rxh5620   34439 Feb 25 13:37 fig4a.ps
-rw-r--r--  1 rxh5620   35602 Mar  3 03:13 fig4a_thesis.ps
-rw-r--r--  1 rxh5620   35692 Mar  3 03:21 fig4b_thesis.ps
-rw-r--r--  1 rxh5620   35419 Mar  3 03:28 fig4c_thesis.ps
-rw-rw-rw-  1 rxh5620   16630 Mar  1 12:48 fig_3_jet.ps
-rw-rw-rw-  1 rxh5620   16630 Mar  1 12:50 fig_3_prism.ps
-rw-rw-rw-  1 rxh5620    2246 Nov 21 17:17 file_index.ascii
-rw-rw-rw-  1 rxh5620     657 Nov 21 17:17 fileget.m
-rw-rw-rw-  1 rxh5620    5497 Nov 21 17:17 filenames
-rw-rw-rw-  1 rxh5620     714 Nov 21 17:17 filesend.m
-rw-rw-rw-  1 rxh5620   42292 Feb 25 02:19 graph1.eps
-rw-rw-rw-  1 rxh5620   10721 Feb 25 02:01 graph1.gif
-rw-rw-rw-  1 rxh5620   66704 Feb 25 02:32 graph1.pcx
-rw-rw-rw-  1 rxh5620   46865 Feb 25 01:45 graph1.ps
-rw-rw-rw-  1 rxh5620   16630 Feb 25 03:29 graph3.ps
-rw-rw-rw-  1 rxh5620   34305 Feb 25 03:03 graph4.ps
-rw-r--r--  1 rxh5620    1463 Dec  5 00:44 i_ave.m
-rw-rw-rw-  1 rxh5620   16416 Mar  5 01:39 image1.ascii
-rw-rw-rw-  1 rxh5620   16416 Mar  5 01:31 image2.ascii
-rw-rw-rw-  1 rxh5620   16416 Mar  3 03:16 image3.ascii
-rw-rw-rw-  1 rxh5620     567 Feb 27 20:44 image32x32.m
-rw-rw-rw-  1 rxh5620     740 Feb 25 11:53 image32x32_1.m
```

```
-rw-rw-rw-  1 rxh5620     741 Feb 25 11:56 image32x32_2.m
-rw-rw-rw-  1 rxh5620     796 Mar  3 03:04 image32x32_3.m
-rw-rw-rw-  1 rxh5620     794 Mar  3 03:18 image32x32_4.m
-rw-rw-rw-  1 rxh5620     796 Mar  3 03:27 image32x32_5.m
-rw-rw-rw-  1 rxh5620   16416 Mar  3 03:19 image4.ascii
-rw-r--r--  1 rxh5620   16416 Mar  3 03:27 image5.ascii
-rw-rw-rw-  1 rxh5620    4112 Nov 21 17:17 image6.ascii
-rw-rw-rw-  1 rxh5620     573 Nov 29 20:24 image_1.m
-rw-rw-rw-  1 rxh5620     733 Nov 30 13:24 image_2.m
-rw-rw-rw-  1 rxh5620     661 Nov 21 17:17 image_3.m
-rw-rw-rw-  1 rxh5620     588 Nov 21 17:17 image_4.m
-rw-rw-rw-  1 rxh5620     652 Nov 21 17:17 image_5.m
-rw-rw-rw-  1 rxh5620     716 Nov 21 17:17 image_6.m
-rw-rw-rw-  1 rxh5620    2362 Nov 21 17:17 image_c_1.ascii
-rw-rw-rw-  1 rxh5620    2356 Nov 21 17:17 image_c_2.ascii
-rw-rw-rw-  1 rxh5620     710 Nov 30 16:12 imagecdfdiff.m
-rw-rw-rw-  1 rxh5620     725 Nov 30 16:11 imagecdfratio.m
-rw-rw-rw-  1 rxh5620     635 Nov 30 13:25 imagediff.m
-rw-rw-rw-  1 rxh5620     736 Nov 30 13:39 imagehistdiff.m
-rw-rw-rw-  1 rxh5620     750 Nov 30 13:40 imagehistratio.m
-rw-rw-rw-  1 rxh5620     648 Nov 30 13:27 imageratio.m
-rw-rw-rw-  1 rxh5620     658 Nov 28 22:36 imagexcdf.m
-rw-rw-rw-  1 rxh5620     769 Nov 28 22:37 imagexhist.m
-rw-rw-rw-  1 rxh5620     642 Nov 28 22:38 imageycdf.m
-rw-rw-rw-  1 rxh5620     782 Nov 28 22:38 imageyhist.m
-rw-rw-rw-  1 rxh5620       0 Nov 21 17:17 main.m
-rw-rw-rw-  1 rxh5620    8654 Dec  1 12:35 main1.m
-rw-rw-rw-  1 rxh5620    8539 Nov 21 17:17 main10.m
-rw-rw-rw-  1 rxh5620    8511 Nov 21 17:18 main2.m
-rw-rw-rw-  1 rxh5620    8648 Feb 25 11:23 main32x32_1.m
-rw-rw-rw-  1 rxh5620    8655 Mar  1 12:31 main32x32_1a.m
-rw-rw-rw-  1 rxh5620    8656 Mar  3 03:24 main32x32_2.m
-rw-r--r--  1 rxh5620    2685 Dec  5 02:17 main_001.m
-rw-r--r--  1 rxh5620    4967 Dec  1 14:26 main_002.m
-rw-rw-rw-  1 rxh5620       0 Nov 21 17:18 main_rotate
-rw-rw-rw-  1 rxh5620    8672 Nov 30 16:36 main_rotate.m
-rw-rw-rw-  1 rxh5620     122 Nov 21 17:18 main_rotate_script
-rw-rw-rw-  1 rxh5620    8581 Nov 30 17:59 main_short.m
-rw-rw-rw-  1 rxh5620       0 Mar  5 01:38 matlab_example
-rw-rw-rw-  1 rxh5620     450 Nov 21 17:18 pfft.m
-rw-rw-rw-  1 rxh5620     387 Nov 21 17:18 rlogin.m
-rw-rw-rw-  1 rxh5620     606 Nov 28 22:40 rms_error.m
-rw-rw-rw-  1 rxh5620     588 Nov 28 22:43 rms_error_1d.m
-rw-rw-rw-  1 rxh5620    1754 Nov 28 22:45 rms_main.m
-rw-r--r--  1 rxh5620    1742 Nov 29 20:36 rotate.m
-rw-rw-rw-  1 rxh5620     685 Nov 21 17:18 run1.m
-rw-r--r--  1 rxh5620   56663 Nov 21 17:18 rundiff.ascii
-rw-r--r--  1 rxh5620     893 Nov 30 21:47 trials_8.m
-rw-rw-rw-  1 rxh5620   25643 Nov 21 17:18 typescript
/afs/cad.njit.edu/usr7/rxh5620/image_64

image1 saved as image1.ascii
```

image_x =

Columns 1 through 12

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Columns 13 through 24

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	63	0	0	0	0	0	0	0	0
0	0	0	63	0	0	0	0	0	0	0	0
0	0	0	63	0	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0
0	0	63	63	63	0	0	0	0	0	0	0

```
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0   63   63   63    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
```

Columns 25 through 32

```
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
```

image name..: image2
description.: vertical line with defect
size.....: 32x32

image1 created

image2 saved as image2.ascii

image_y =

Columns 1 through 12

```
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0    63    63    63    63    63
     0     0     0     0    63    63    63    63    63    63    63    63
     0     0     0     0     0     0     0    63    63    63    63    63
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

Columns 13 through 24

```
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
   63   63   63   63   63   63   63   63   63   63   63   63
   63   63   63   63   63   63   63   63   63   63   63   63
   63   63   63   63   63   63   63   63   63   63   63   63
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0
```

Columns 25 through 32

```
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
```

```
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
    63    63    63    63     0     0     0     0
    63    63    63    63     0     0     0     0
    63    63    63    63     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0

---------------------------------------
image_x created
---------------------------------------
surface plot of image_x created
---------------------------------------
image_y created
---------------------------------------
surface plot of image_y created
---------------------------------------
calculating image_y-image_x
---------------------------------------
image_diff created
---------------------------------------
image_diff_min =
   -63

---------------------------------------
image_diff_max =
    63

---------------------------------------
end of image_diff
---------------------------------------
surface plot of image_diff
---------------------------------------
calculating image_y/image_x
---------------------------------------

Warning: Divide by zero
image_ratio created
---------------------------------------
image_ratio_min =
   NaN

---------------------------------------
image_ratio_max =
   NaN

---------------------------------------
end of image_ratio
---------------------------------------
surface plot of image_ratio
---------------------------------------
fftx created
---------------------------------------
```

```
minimum value of fftx =
    0

---------------------------------------
maximum value of fftx =
      4158

---------------------------------------

fftxabs created
---------------------------------------
minimum value of fftxabs =
    0

---------------------------------------
maximum value of fftxabs =
      4158


---------------------------------------
fftxshift created
---------------------------------------
minimum value of fftxshift =
    0

---------------------------------------
maximum value of fftxshift =
      4158


Warning: Log of zero
---------------------------------------
fftxlog created
---------------------------------------
minimum value of fftxlog =
  -Inf

---------------------------------------
maximum value of fftxlog =
   4.6189


---------------------------------------
fftxmax created


---------------------------------------
ffxscaled created
---------------------------------------
minimum value of fftxscaled =
  -Inf

---------------------------------------
```

```
maximum value of fftxscaled =
   32

---------------------------------------
end of fft process
---------------------------------------
ffty created
---------------------------------------
minimum value of ffty =
    0

---------------------------------------
maximum value of ffty =
      4158


---------------------------------------
fftyabs created
---------------------------------------
minimum value of fftyabs =
    0

---------------------------------------
maximum value of fftyabs =
      4158


---------------------------------------
fftyshift created
---------------------------------------
minimum value of fftyshift =
    0

---------------------------------------
maximum value of fftyshift =
      4158


Warning: Log of zero
---------------------------------------
fftylog created
---------------------------------------
minimum value of fftylog =
 -Inf

---------------------------------------
maximum value of fftylog =
   4.6189


---------------------------------------
fftymax created
```

```
---------------------------------------
fftyscaled created
---------------------------------------
minimum value of fftyscaled =
  -Inf

---------------------------------------
maximum value of fftyscaled =
    32

---------------------------------------
end of fft process
---------------------------------------
fft of image_x created
---------------------------------------
surfc of fftxscaled created
---------------------------------------
fft of image_y created
---------------------------------------
surfc of fftyscaled created
---------------------------------------
calculating ffty_def-fftxdef
---------------------------------------
fft_diff created
---------------------------------------
fft_diff_min =
   NaN

---------------------------------------
fft_diff_max =
   NaN

---------------------------------------
end of fft_diff
---------------------------------------
surface plot of fft_diff
---------------------------------------
calculating ffty_def/fftxdef
---------------------------------------
fft_ratio created
---------------------------------------
fft_ratio_min =
   NaN

---------------------------------------
fft_ratio_max =
   NaN

---------------------------------------
end of fft_ratio
---------------------------------------
surface plot of fft_ratio
```

fft_abs_diff =

1.0e+03 *

Columns 1 through 7

0	-2.5330	-3.1313	-3.5814	-3.0788	-2.7466	-2.3951
2.5330	0	-0.6730	-1.2528	-0.9452	-0.8411	-0.7496
3.1313	0.6730	0	-0.6051	-0.3916	-0.3734	-0.3790
3.5814	1.2528	0.6051	0.0000	0.1166	0.0644	-0.0178
3.0788	0.9452	0.3916	-0.1166	0	0.0126	0.0070
2.7466	0.8411	0.3734	-0.0644	-0.0126	-0.0000	-0.0084
2.3951	0.7496	0.3790	0.0178	-0.0070	0.0084	0.0000
1.7966	0.4411	0.1965	-0.0526	-0.1192	-0.0456	-0.0072
1.3860	0.3309	0.2065	0.0487	-0.0891	0.0245	0.0652
0.8940	0.1502	0.1445	0.0647	-0.0292	0.0173	0.0379
0.2760	-0.0886	-0.0220	-0.0342	-0.0636	-0.0405	-0.0207
-0.0990	0.0961	0.0478	0.0437	0.0545	0.0406	0.0261
0.3068	0.3385	0.1629	0.1307	0.1782	0.1370	0.0916
0.5185	0.3574	0.0929	0.0301	0.1209	0.0751	0.0211
0.8129	0.5081	0.1750	0.0720	0.1797	0.1122	0.0302
1.0198	0.6272	0.2543	0.1264	0.2465	0.1677	0.0693
1.0080	0.5875	0.2065	0.0787	0.2151	0.1486	0.0652
1.0198	0.6272	0.2543	0.1264	0.2465	0.1677	0.0693
0.8129	0.5081	0.1750	0.0720	0.1797	0.1122	0.0302
0.5185	0.3574	0.0929	0.0301	0.1209	0.0751	0.0211
0.3068	0.3385	0.1629	0.1307	0.1782	0.1370	0.0916
-0.0990	0.0961	0.0478	0.0437	0.0545	0.0406	0.0261
0.2760	-0.0886	-0.0220	-0.0342	-0.0636	-0.0405	-0.0207
0.8940	0.1502	0.1445	0.0647	-0.0292	0.0173	0.0379
1.3860	0.3309	0.2065	0.0487	-0.0891	0.0245	0.0652
1.7966	0.4411	0.1965	-0.0526	-0.1192	-0.0456	-0.0072
2.3951	0.7496	0.3790	0.0178	-0.0070	0.0084	0
2.7466	0.8411	0.3734	-0.0644	-0.0126	-0.0000	-0.0084
3.0788	0.9452	0.3916	-0.1166	0	0.0126	0.0070
3.5814	1.2528	0.6051	0.0000	0.1166	0.0644	-0.0178
3.1313	0.6730	-0.0000	-0.6051	-0.3916	-0.3734	-0.3790
2.5330	0	-0.6730	-1.2528	-0.9452	-0.8411	-0.7496

Columns 8 through 14

-1.7966	-1.3860	-0.8940	-0.2760	0.0990	-0.3068	-0.5185
-0.4411	-0.3309	-0.1502	0.0886	-0.0961	-0.3385	-0.3574
-0.1965	-0.2065	-0.1445	0.0220	-0.0478	-0.1629	-0.0929
0.0526	-0.0487	-0.0647	0.0342	-0.0437	-0.1307	-0.0301
0.1192	0.0891	0.0292	0.0636	-0.0545	-0.1782	-0.1209
0.0456	-0.0245	-0.0173	0.0405	-0.0406	-0.1370	-0.0751
0.0072	-0.0652	-0.0379	0.0207	-0.0261	-0.0916	-0.0211
0.0000	-0.0456	0.0039	0.0308	-0.0272	-0.1085	-0.0712
0.0456	0	0.0330	0.0276	-0.0195	-0.0891	-0.0582
-0.0039	-0.0330	-0.0000	0.0051	-0.0036	-0.0221	0.0018
-0.0308	-0.0276	-0.0051	0.0000	-0.0006	-0.0142	-0.0187
0.0272	0.0195	0.0036	0.0006	-0.0000	0.0143	0.0216
0.1085	0.0891	0.0221	0.0142	-0.0143	0	0.0205

0.0712	0.0582	-0.0018	0.0187	-0.0216	-0.0205	0
0.0825	0.0522	-0.0296	0.0106	-0.0197	-0.0072	0.0316
0.1225	0.0788	-0.0219	0.0171	-0.0240	-0.0187	0.0213
0.1442	0.1260	0.0028	0.0276	-0.0293	-0.0369	-0.0052
0.1225	0.0788	-0.0219	0.0171	-0.0240	-0.0187	0.0213
0.0825	0.0522	-0.0296	0.0106	-0.0197	-0.0072	0.0316
0.0712	0.0582	-0.0018	0.0187	-0.0216	-0.0205	-0.0000
0.1085	0.0891	0.0221	0.0142	-0.0143	-0.0000	0.0205
0.0272	0.0195	0.0036	0.0006	-0.0000	0.0143	0.0216
-0.0308	-0.0276	-0.0051	0.0000	-0.0006	-0.0142	-0.0187
-0.0039	-0.0330	-0.0000	0.0051	-0.0036	-0.0221	0.0018
0.0456	0	0.0330	0.0276	-0.0195	-0.0891	-0.0582
0.0000	-0.0456	0.0039	0.0308	-0.0272	-0.1085	-0.0712
0.0072	-0.0652	-0.0379	0.0207	-0.0261	-0.0916	-0.0211
0.0456	-0.0245	-0.0173	0.0405	-0.0406	-0.1370	-0.0751
0.1192	0.0891	0.0292	0.0636	-0.0545	-0.1782	-0.1209
0.0526	-0.0487	-0.0647	0.0342	-0.0437	-0.1307	-0.0301
-0.1965	-0.2065	-0.1445	0.0220	-0.0478	-0.1629	-0.0929
-0.4411	-0.3309	-0.1502	0.0886	-0.0961	-0.3385	-0.3574

Columns 15 through 21

-0.8129	-1.0198	-1.0080	-1.0198	-0.8129	-0.5185	-0.3068
-0.5081	-0.6272	-0.5875	-0.6272	-0.5081	-0.3574	-0.3385
-0.1750	-0.2543	-0.2065	-0.2543	-0.1750	-0.0929	-0.1629
-0.0720	-0.1264	-0.0787	-0.1264	-0.0720	-0.0301	-0.1307
-0.1797	-0.2465	-0.2151	-0.2465	-0.1797	-0.1209	-0.1782
-0.1122	-0.1677	-0.1486	-0.1677	-0.1122	-0.0751	-0.1370
-0.0302	-0.0693	-0.0652	-0.0693	-0.0302	-0.0211	-0.0916
-0.0825	-0.1225	-0.1442	-0.1225	-0.0825	-0.0712	-0.1085
-0.0522	-0.0788	-0.1260	-0.0788	-0.0522	-0.0582	-0.0891
0.0296	0.0219	-0.0028	0.0219	0.0296	0.0018	-0.0221
-0.0106	-0.0171	-0.0276	-0.0171	-0.0106	-0.0187	-0.0142
0.0197	0.0240	0.0293	0.0240	0.0197	0.0216	0.0143
0.0072	0.0187	0.0369	0.0187	0.0072	0.0205	0.0000
-0.0316	-0.0213	0.0052	-0.0213	-0.0316	0.0000	-0.0205
-0.0000	0.0170	0.0522	0.0170	-0.0000	0.0316	-0.0072
-0.0170	-0.0000	0.0392	0.0000	-0.0170	0.0213	-0.0187
-0.0522	-0.0392	0	-0.0392	-0.0522	-0.0052	-0.0369
-0.0170	-0.0000	0.0392	0.0000	-0.0170	0.0213	-0.0187
-0.0000	0.0170	0.0522	0.0170	-0.0000	0.0316	-0.0072
-0.0316	-0.0213	0.0052	-0.0213	-0.0316	-0.0000	-0.0205
0.0072	0.0187	0.0369	0.0187	0.0072	0.0205	0.0000
0.0197	0.0240	0.0293	0.0240	0.0197	0.0216	0.0143
-0.0106	-0.0171	-0.0276	-0.0171	-0.0106	-0.0187	-0.0142
0.0296	0.0219	-0.0028	0.0219	0.0296	0.0018	-0.0221
-0.0522	-0.0788	-0.1260	-0.0788	-0.0522	-0.0582	-0.0891
-0.0825	-0.1225	-0.1442	-0.1225	-0.0825	-0.0712	-0.1085
-0.0302	-0.0693	-0.0652	-0.0693	-0.0302	-0.0211	-0.0916
-0.1122	-0.1677	-0.1486	-0.1677	-0.1122	-0.0751	-0.1370
-0.1797	-0.2465	-0.2151	-0.2465	-0.1797	-0.1209	-0.1782
-0.0720	-0.1264	-0.0787	-0.1264	-0.0720	-0.0301	-0.1307
-0.1750	-0.2543	-0.2065	-0.2543	-0.1750	-0.0929	-0.1629
-0.5081	-0.6272	-0.5875	-0.6272	-0.5081	-0.3574	-0.3385

Columns 22 through 28

0.0990	-0.2760	-0.8940	-1.3860	-1.7966	-2.3951	-2.7466
-0.0961	0.0886	-0.1502	-0.3309	-0.4411	-0.7496	-0.8411
-0.0478	0.0220	-0.1445	-0.2065	-0.1965	-0.3790	-0.3734
-0.0437	0.0342	-0.0647	-0.0487	0.0526	-0.0178	0.0644
-0.0545	0.0636	0.0292	0.0891	0.1192	0.0070	0.0126
-0.0406	0.0405	-0.0173	-0.0245	0.0456	-0.0084	0.0000
-0.0261	0.0207	-0.0379	-0.0652	0.0072	-0.0000	0.0084
-0.0272	0.0308	0.0039	-0.0456	0.0000	-0.0072	-0.0456
-0.0195	0.0276	0.0330	0	0.0456	0.0652	0.0245
-0.0036	0.0051	0.0000	-0.0330	-0.0039	0.0379	0.0173
-0.0006	0.0000	-0.0051	-0.0276	-0.0308	-0.0207	-0.0405
0.0000	0.0006	0.0036	0.0195	0.0272	0.0261	0.0406
-0.0143	0.0142	0.0221	0.0891	0.1085	0.0916	0.1370
-0.0216	0.0187	-0.0018	0.0582	0.0712	0.0211	0.0751
-0.0197	0.0106	-0.0296	0.0522	0.0825	0.0302	0.1122
-0.0240	0.0171	-0.0219	0.0788	0.1225	0.0693	0.1677
-0.0293	0.0276	0.0028	0.1260	0.1442	0.0652	0.1486
-0.0240	0.0171	-0.0219	0.0788	0.1225	0.0693	0.1677
-0.0197	0.0106	-0.0296	0.0522	0.0825	0.0302	0.1122
-0.0216	0.0187	-0.0018	0.0582	0.0712	0.0211	0.0751
-0.0143	0.0142	0.0221	0.0891	0.1085	0.0916	0.1370
-0.0000	0.0006	0.0036	0.0195	0.0272	0.0261	0.0406
-0.0006	0.0000	-0.0051	-0.0276	-0.0308	-0.0207	-0.0405
-0.0036	0.0051	0	-0.0330	-0.0039	0.0379	0.0173
-0.0195	0.0276	0.0330	0	0.0456	0.0652	0.0245
-0.0272	0.0308	0.0039	-0.0456	0.0000	-0.0072	-0.0456
-0.0261	0.0207	-0.0379	-0.0652	0.0072	0	0.0084
-0.0406	0.0405	-0.0173	-0.0245	0.0456	-0.0084	-0.0000
-0.0545	0.0636	0.0292	0.0891	0.1192	0.0070	0.0126
-0.0437	0.0342	-0.0647	-0.0487	0.0526	-0.0178	0.0644
-0.0478	0.0220	-0.1445	-0.2065	-0.1965	-0.3790	-0.3734
-0.0961	0.0886	-0.1502	-0.3309	-0.4411	-0.7496	-0.8411

Columns 29 through 32

-3.0788	-3.5814	-3.1313	-2.5330
-0.9452	-1.2528	-0.6730	0
-0.3916	-0.6051	-0.0000	0.6730
0.1166	-0.0000	0.6051	1.2528
-0.0000	-0.1166	0.3916	0.9452
-0.0126	-0.0644	0.3734	0.8411
-0.0070	0.0178	0.3790	0.7496
-0.1192	-0.0526	0.1965	0.4411
-0.0891	0.0487	0.2065	0.3309
-0.0292	0.0647	0.1445	0.1502
-0.0636	-0.0342	-0.0220	-0.0886
0.0545	0.0437	0.0478	0.0961
0.1782	0.1307	0.1629	0.3385
0.1209	0.0301	0.0929	0.3574
0.1797	0.0720	0.1750	0.5081
0.2465	0.1264	0.2543	0.6272

```
  0.2151   0.0787   0.2065   0.5875
  0.2465   0.1264   0.2543   0.6272
  0.1797   0.0720   0.1750   0.5081
  0.1209   0.0301   0.0929   0.3574
  0.1782   0.1307   0.1629   0.3385
  0.0545   0.0437   0.0478   0.0961
 -0.0636  -0.0342  -0.0220  -0.0886
 -0.0292   0.0647   0.1445   0.1502
 -0.0891   0.0487   0.2065   0.3309
 -0.1192  -0.0526   0.1965   0.4411
 -0.0070   0.0178   0.3790   0.7496
 -0.0126  -0.0644   0.3734   0.8411
 -0.0000  -0.1166   0.3916   0.9452
  0.1166  -0.0000   0.6051   1.2528
 -0.3916  -0.6051  -0.0000   0.6730
 -0.9452  -1.2528  -0.6730   0.0000

---------------------------------------
surface plot of fft_abs_diff

Warning: Divide by zero

fft_abs_ratio =

 Columns 1 through 7

  1.0000   0.3833   0.2086   0.0352   0.0899   0.0790   0.0513
  2.6091   1.0000   0.5478   0.0947   0.2399   0.2161   0.1469
  4.7944   1.8256   1.0000   0.1792   0.4178   0.3719   0.2522
 28.4062  10.5620   5.5817   1.0000   1.8546   1.4608   0.8759
 11.1213   4.1683   2.3933   0.5392   1.0000   1.0746   1.0599
 12.6638   4.6278   2.6885   0.6846   0.9306   1.0000   0.9349
 19.4925   6.8075   3.9651   1.1417   0.9435   1.0696   1.0000
  8.7574   2.9342   1.9024   0.7383   0.3323   0.6999   0.9410
 12.0000   3.6777   2.7741   1.4648        0   1.3500   2.3518
  9.7789   2.4881   2.4693   1.6869   0.6702   1.2114   1.5121
  2.2352   0.5981   0.8963   0.8278   0.6467   0.7433   0.8433
  0.2976   1.6903   1.3563   1.3477   1.4764   1.4031   1.3073
  6.8787   7.6138   4.3777   4.0121   5.8284   5.7264   5.5874
  4.0567   3.1387   1.5815   1.2035   1.9163   1.6659   1.2320
  7.7948   5.2909   2.5245   1.6601   2.7725   2.2147   1.3656
 17.0946  11.0318   5.2319   3.2433   5.7544   4.5421   2.5763
  9.0000   5.7544   2.7741   1.7508   3.4142   3.1230   2.3518
 17.0946  11.0318   5.2319   3.2433   5.7544   4.5421   2.5763
  7.7948   5.2909   2.5245   1.6601   2.7725   2.2147   1.3656
  4.0567   3.1387   1.5815   1.2035   1.9163   1.6659   1.2320
  6.8787   7.6138   4.3777   4.0121   5.8284   5.7264   5.5874
  0.2976   1.6903   1.3563   1.3477   1.4764   1.4031   1.3073
  2.2352   0.5981   0.8963   0.8278   0.6467   0.7433   0.8433
  9.7789   2.4881   2.4693   1.6869   0.6702   1.2114   1.5121
 12.0000   3.6777   2.7741   1.4648        0   1.3500   2.3518
  8.7574   2.9342   1.9024   0.7383   0.3323   0.6999   0.9410
 19.4925   6.8075   3.9651   1.1417   0.9435   1.0696   1.0000
 12.6638   4.6278   2.6885   0.6846   0.9306   1.0000   0.9349
```

11.1213	4.1683	2.3933	0.5392	1.0000	1.0746	1.0599
28.4062	10.5620	5.5817	1.0000	1.8546	1.4608	0.8759
4.7944	1.8256	1.0000	0.1792	0.4178	0.3719	0.2522
2.6091	1.0000	0.5478	0.0947	0.2399	0.2161	0.1469

Columns 8 through 14

0.1142	0.0833	0.1023	0.4474	3.3600	0.1454	0.2465
0.3408	0.2719	0.4019	1.6719	0.5916	0.1313	0.3186
0.5256	0.3605	0.4050	1.1157	0.7373	0.2284	0.6323
1.3545	0.6827	0.5928	1.2080	0.7420	0.2492	0.8309
3.0092	Inf	1.4922	1.5464	0.6773	0.1716	0.5218
1.4288	0.7407	0.8255	1.3454	0.7127	0.1746	0.6003
1.0627	0.4252	0.6613	1.1858	0.7650	0.1790	0.8117
1.0000	0.3501	1.0621	1.4194	0.7146	0.0858	0.4896
2.8567	NaN	2.3444	1.5714	0.7216	0	0.4443
0.9415	0.4265	1.0000	1.1209	0.9004	0.3158	1.0619
0.7045	0.6364	0.8921	1.0000	0.9657	0.5849	0.6368
1.3994	1.3858	1.1106	1.0355	1.0000	1.9751	1.8453
11.6603	Inf	3.1667	1.7097	0.5063	1.0000	1.4726
2.0425	2.2505	0.9417	1.5705	0.5419	0.6791	1.0000
2.1314	1.8123	0.4869	1.1970	0.6317	0.8701	1.5417
3.8580	2.7607	0.5570	1.3066	0.6184	0.7326	1.2803
6.8644	Inf	1.1143	1.5714	0.5810	0.5858	0.9501
3.8580	2.7607	0.5570	1.3066	0.6184	0.7326	1.2803
2.1314	1.8123	0.4869	1.1970	0.6317	0.8701	1.5417
2.0425	2.2505	0.9417	1.5705	0.5419	0.6791	1.0000
11.6603	Inf	3.1667	1.7097	0.5063	1.0000	1.4726
1.3994	1.3858	1.1106	1.0355	1.0000	1.9751	1.8453
0.7045	0.6364	0.8921	1.0000	0.9657	0.5849	0.6368
0.9415	0.4265	1.0000	1.1209	0.9004	0.3158	1.0619
2.8567	NaN	2.3444	1.5714	0.7216	0	0.4443
1.0000	0.3501	1.0621	1.4194	0.7146	0.0858	0.4896
1.0627	0.4252	0.6613	1.1858	0.7650	0.1790	0.8117
1.4288	0.7407	0.8255	1.3454	0.7127	0.1746	0.6003
3.0092	Inf	1.4922	1.5464	0.6773	0.1716	0.5218
1.3545	0.6827	0.5928	1.2080	0.7420	0.2492	0.8309
0.5256	0.3605	0.4050	1.1157	0.7373	0.2284	0.6323
0.3408	0.2719	0.4019	1.6719	0.5916	0.1313	0.3186

Columns 15 through 21

0.1283	0.0585	0.1111	0.0585	0.1283	0.2465	0.1454
0.1890	0.0906	0.1738	0.0906	0.1890	0.3186	0.1313
0.3961	0.1911	0.3605	0.1911	0.3961	0.6323	0.2284
0.6024	0.3083	0.5712	0.3083	0.6024	0.8309	0.2492
0.3607	0.1738	0.2929	0.1738	0.3607	0.5218	0.1716
0.4515	0.2202	0.3202	0.2202	0.4515	0.6003	0.1746
0.7323	0.3882	0.4252	0.3882	0.7323	0.8117	0.1790
0.4692	0.2592	0.1457	0.2592	0.4692	0.4896	0.0858
0.5518	0.3622	0	0.3622	0.5518	0.4443	0
2.0538	1.7954	0.8974	1.7954	2.0538	1.0619	0.3158
0.8354	0.7653	0.6364	0.7653	0.8354	0.6368	0.5849
1.5830	1.6170	1.7212	1.6170	1.5830	1.8453	1.9751

1.1493	1.3649	1.7071	1.3649	1.1493	1.4726	1.0000
0.6486	0.7811	1.0526	0.7811	0.6486	1.0000	0.6791
1.0000	1.2678	1.8123	1.2678	1.0000	1.5417	0.8701
0.7888	1.0000	1.4649	1.0000	0.7888	1.2803	0.7326
0.5518	0.6827	1.0000	0.6827	0.5518	0.9501	0.5858
0.7888	1.0000	1.4649	1.0000	0.7888	1.2803	0.7326
1.0000	1.2678	1.8123	1.2678	1.0000	1.5417	0.8701
0.6486	0.7811	1.0526	0.7811	0.6486	1.0000	0.6791
1.1493	1.3649	1.7071	1.3649	1.1493	1.4726	1.0000
1.5830	1.6170	1.7212	1.6170	1.5830	1.8453	1.9751
0.8354	0.7653	0.6364	0.7653	0.8354	0.6368	0.5849
2.0538	1.7954	0.8974	1.7954	2.0538	1.0619	0.3158
0.5518	0.3622	0	0.3622	0.5518	0.4443	0
0.4692	0.2592	0.1457	0.2592	0.4692	0.4896	0.0858
0.7323	0.3882	0.4252	0.3882	0.7323	0.8117	0.1790
0.4515	0.2202	0.3202	0.2202	0.4515	0.6003	0.1746
0.3607	0.1738	0.2929	0.1738	0.3607	0.5218	0.1716
0.6024	0.3083	0.5712	0.3083	0.6024	0.8309	0.2492
0.3961	0.1911	0.3605	0.1911	0.3961	0.6323	0.2284
0.1890	0.0906	0.1738	0.0906	0.1890	0.3186	0.1313

Columns 22 through 28

3.3600	0.4474	0.1023	0.0833	0.1142	0.0513	0.0790
0.5916	1.6719	0.4019	0.2719	0.3408	0.1469	0.2161
0.7373	1.1157	0.4050	0.3605	0.5256	0.2522	0.3719
0.7420	1.2080	0.5928	0.6827	1.3545	0.8759	1.4608
0.6773	1.5464	1.4922	Inf	3.0092	1.0599	1.0746
0.7127	1.3454	0.8255	0.7407	1.4288	0.9349	1.0000
0.7650	1.1858	0.6613	0.4252	1.0627	1.0000	1.0696
0.7146	1.4194	1.0621	0.3501	1.0000	0.9410	0.6999
0.7216	1.5714	2.3444	NaN	2.8567	2.3518	1.3500
0.9004	1.1209	1.0000	0.4265	0.9415	1.5121	1.2114
0.9657	1.0000	0.8921	0.6364	0.7045	0.8433	0.7433
1.0000	1.0355	1.1106	1.3858	1.3994	1.3073	1.4031
0.5063	1.7097	3.1667	Inf	11.6603	5.5874	5.7264
0.5419	1.5705	0.9417	2.2505	2.0425	1.2320	1.6659
0.6317	1.1970	0.4869	1.8123	2.1314	1.3656	2.2147
0.6184	1.3066	0.5570	2.7607	3.8580	2.5763	4.5421
0.5810	1.5714	1.1143	Inf	6.8644	2.3518	3.1230
0.6184	1.3066	0.5570	2.7607	3.8580	2.5763	4.5421
0.6317	1.1970	0.4869	1.8123	2.1314	1.3656	2.2147
0.5419	1.5705	0.9417	2.2505	2.0425	1.2320	1.6659
0.5063	1.7097	3.1667	Inf	11.6603	5.5874	5.7264
1.0000	1.0355	1.1106	1.3858	1.3994	1.3073	1.4031
0.9657	1.0000	0.8921	0.6364	0.7045	0.8433	0.7433
0.9004	1.1209	1.0000	0.4265	0.9415	1.5121	1.2114
0.7216	1.5714	2.3444	NaN	2.8567	2.3518	1.3500
0.7146	1.4194	1.0621	0.3501	1.0000	0.9410	0.6999
0.7650	1.1858	0.6613	0.4252	1.0627	1.0000	1.0696
0.7127	1.3454	0.8255	0.7407	1.4288	0.9349	1.0000
0.6773	1.5464	1.4922	Inf	3.0092	1.0599	1.0746
0.7420	1.2080	0.5928	0.6827	1.3545	0.8759	1.4608
0.7373	1.1157	0.4050	0.3605	0.5256	0.2522	0.3719

```
  0.5916   1.6719   0.4019   0.2719   0.3408   0.1469   0.2161

 Columns 29 through 32

  0.0899   0.0352   0.2086   0.3833
  0.2399   0.0947   0.5478   1.0000
  0.4178   0.1792   1.0000   1.8256
  1.8546   1.0000   5.5817  10.5620
  1.0000   0.5392   2.3933   4.1683
  0.9306   0.6846   2.6885   4.6278
  0.9435   1.1417   3.9651   6.8075
  0.3323   0.7383   1.9024   2.9342
     0   1.4648   2.7741   3.6777
  0.6702   1.6869   2.4693   2.4881
  0.6467   0.8278   0.8963   0.5981
  1.4764   1.3477   1.3563   1.6903
  5.8284   4.0121   4.3777   7.6138
  1.9163   1.2035   1.5815   3.1387
  2.7725   1.6601   2.5245   5.2909
  5.7544   3.2433   5.2319  11.0318
  3.4142   1.7508   2.7741   5.7544
  5.7544   3.2433   5.2319  11.0318
  2.7725   1.6601   2.5245   5.2909
  1.9163   1.2035   1.5815   3.1387
  5.8284   4.0121   4.3777   7.6138
  1.4764   1.3477   1.3563   1.6903
  0.6467   0.8278   0.8963   0.5981
  0.6702   1.6869   2.4693   2.4881
     0   1.4648   2.7741   3.6777
  0.3323   0.7383   1.9024   2.9342
  0.9435   1.1417   3.9651   6.8075
  0.9306   0.6846   2.6885   4.6278
  1.0000   0.5392   2.3933   4.1683
  1.8546   1.0000   5.5817  10.5620
  0.4178   0.1792   1.0000   1.8256
  0.2399   0.0947   0.5478   1.0000

----------------------------------------
surface plot of fft_abs_ratio
----------------------------------------
image_x created
----------------------------------------
image_y created
----------------------------------------
fft of image_x created
----------------------------------------
fft of image_y created
----------------------------------------
start histogram calculation of image_x
----------------------------------------
matrix size  =
   32

----------------------------------------
```

```
image_x_hist created
--------------------------------------
end of histogram routine
--------------------------------------
start histogram calculation of image_y
--------------------------------------
matrix size  =
   32

--------------------------------------
image_y_hist created
--------------------------------------
end of histogram routine
plot of image_x_hist created
----------------------------------------
plot of image_y_hist created
----------------------------------------
calculating image_y_hist-image_x_hist
----------------------------------------
image_hist_diff created
----------------------------------------
image_hist_diff_min =
    0

----------------------------------------
image_hist_diff_max =
    0

----------------------------------------
end of image_hist_diff
----------------------------------------
calculating image_y_hist./image_x_hist
----------------------------------------

Warning: Divide by zero
image_hist_ratio created
----------------------------------------
image_hist_ratio_min =
  NaN

----------------------------------------
image_hist_ratio_max =
  NaN

----------------------------------------
end of image_hist_ratio
----------------------------------------
plot of image_hist_diff created
----------------------------------------
plot of image_hist_ratio created
```

```
--------------------------------------
start histogram calculation of image_x
--------------------------------------
matrix size  =
   32

--------------------------------------
image_x_cdf created
--------------------------------------
end of cdf routine
---------------------------------
start cdf  calculation of image_y
---------------------------------
matrix size  =
   32

--------------------------------------

image_y_cdf =

  958
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
    0
```

```
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0

image_y_cdf created
-------------------------------------
end of cdf routine
-----------------------------------------
plot of image_x_cdf created
-----------------------------------------
plot of image_y_cdf created
-----------------------------------------
calculating image_y_cdf-image_x_cdf
-----------------------------------------
image_cdf_diff created
-----------------------------------------
image_cdf_diff_min =
     0

-----------------------------------------
image_cdf_diff_max =
     0

-----------------------------------------
end of image_cdf_diff
```

```
---------------------------------------
calculating image_y_cdf./image_x_cdf
---------------------------------------
image_cdf_ratio created
---------------------------------------
image_cdf_ratio_min =
     1

---------------------------------------
image_cdf_ratio_max =
     1

---------------------------------------
end of image_cdf_ratio
---------------------------------------
plot of image_cdf_diff created
---------------------------------------
plot of image_cdf_ratio created
---------------------------------------
start histogram calculation of fft_x
---------------------------------------
matrix size  =
    32

---------------------------------------
fft_x_hist created
---------------------------------------
end of histogram routine
---------------------------------------
start histogram calculation of fft_y
---------------------------------------
matrix size  =
    32

---------------------------------------
fft_y_hist created
---------------------------------------
end of histogram routine
plot of fft_x_hist created
---------------------------------------
plot of fft_y_hist created
---------------------------------------
calculating fft_y_hist-fft_x_hist
---------------------------------------
fft_hist_diff created
---------------------------------------
fft_hist_diff_min =
     0

---------------------------------------
fft_hist_diff_max =
     0

---------------------------------------
```

```
end of fft_hist_diff
----------------------------------------
calculating fft_y_hist./fft_x_hist
----------------------------------------

Warning: Divide by zero

fft_hist_ratio =

    1
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
    1
  NaN
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
    1
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
```

```
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN
  NaN

fft_hist_ratio created
----------------------------------------
fft_hist_ratio_min =
  NaN

----------------------------------------
fft_hist_ratio_max =
  NaN

----------------------------------------
end of fft_ratio_diff
----------------------------------------
plot of fft_hist_diff created
----------------------------------------
plot of fft_hist_ratio created
----------------------------------------
start histogram calculation of fft_x
----------------------------------------
matrix size  =
   32

----------------------------------------
fft_x_cdf created
----------------------------------------
end of cdf routine
----------------------------------------
start cdf calculation of fft_y
----------------------------------------
matrix size  =
   32

----------------------------------------
fft_y_cdf created
```

```
--------------------------------------
end of cdf routine
----------------------------------------
plot of fft_x_cdf created
----------------------------------------
plot of fft_y_hist created
----------------------------------------
calculating fft_y_cdf-fft_x_cdf
----------------------------------------

fft_cdf_diff =

     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

```
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0

fft_cdf_diff created
----------------------------------------
fft_cdf_diff_min =
     0

----------------------------------------
fft_cdf_diff_max =
     0

----------------------------------------
end of fft_cdf_diff
----------------------------------------
calculating fft_y_cdf./fft_x_cdf
----------------------------------------
fft_cdf_ratio created
----------------------------------------
fft_cdf_ratio_min =
     1

----------------------------------------
fft_cdf_ratio_max =
     1

----------------------------------------
end of fft_cdf_ratio
----------------------------------------
plot of fft_cdf_diff created
----------------------------------------
plot of fft_cdf_ratio created
```

```
----------------------------------------
Image Error
===========

image_error =

  21.5666


image_snr =

   0.5500

----------------------------------------
Image Histogram Error
=====================

Warning: Divide by zero

image_hist_error =

    0


image_hist_snr =

   Inf

----------------------------------------
Image CDF Error
===============

Warning: Divide by zero

image_cdf_error =

    0


image_cdf_snr =

   Inf

----------------------------------------
FFT Error
=========

fftabs_error =

  540.0135
```

```
fftabs_snr =

    0.8983


fftscaled_error =

    4.2245


fftscaled_snr =

   25.4552

----------------------------------------
FFT Histogram Error
====================

Warning: Divide by zero

fft_hist_error =

     0


fft_hist_snr =

   Inf

----------------------------------------
FFT CDF Error
=============

Warning: Divide by zero

fft_cdf_error =

     0


fft_cdf_snr =

   Inf

%---------------------------------------
fft_hist_error - image_hist_error

ans =

     0

%---------------------------------------
fft_hist_snr   - image_hist_snr
```

```
ans =

  NaN

%--------------------------------------
fft_cdf_error  - image_cdf_error

ans =

    0

%--------------------------------------
fft_cdf_snr    - image_cdf_snr

ans =

  NaN

%--------------------------------------
echo off
>> exit

 258481 flops.

fugue% exit
fugue%
script done on Sun Mar  5 01:42:08 1995
```

APPENDIX C

SEQUENTIAL AND PARALLEL PROGRAM CODE FOR THE TUBONET INCLUDING MEMORY MAPPING

C.1 MEMORY MAPPING

The following block diagram shows the memory location of the program code and data within the Hydra board local and global SRAMS (figure C.1) and Table 5 shows the memory mapping of the Hydra board.

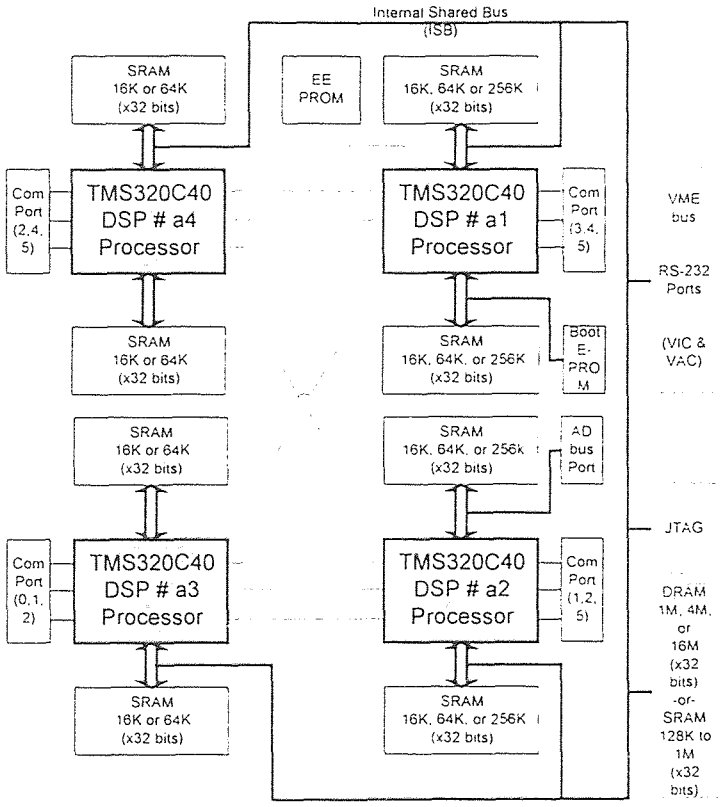

Figure C.1 Block Diagram of Hydra Board.

TABLE 5 Memory Mapping

	Memory Segment	Starting Address	Ending Address	Comment
0	EPROM dspl	30 0000 h	3f ffff h	EPROM
1	not used	40 0000 h	3fff ffff h	
2	Monitor Code	4000 0000 h	4000 0dff h	
3	Interrupt Table	4000 0e00 h	4000 0fff h	
4	Monitor Stack	4000 1000 h	4000 11ff h	
5	Local SRAM 59.5 k	4000 1200 h	4000 ffff h	program code see example run
6	not used	4001 0000 h	8cff ffff h	
7	Global DRAM 1020 k	8d00 0000h	8d0f ffbf h	flag tables see example run
8	DSP Comm. area	8d0f efc0 h	8d0f ffff h	
9	not used	8d10 0000 h	bfff ffff h	
10	Global SRAM 64 k	c000 0000 h	c000 ffff h	data (images. FFTs. Hist..CDFs. etc.)

C.2 PROGRAM CODE FOR SEQUENTIAL CASE (ONE DSP) [8]

Host Program (C code):

```
/*      Host side program  for ONE DSP
*/

#include <stdio.h>
#include <math.h>
#include <signal.h>
#include "vc40dsp.h"
#include <time.h>
/*
                ------------------------------------------------
----------------------------
                ------------------------------------------------
*/

/*

                ------------------------------------------------
---------------------------- Define Host Variables
                ------------------------------------------------
```

```
*/

#define NUMSYMS       (sizeof(symnames)/sizeof(char *))
#define SIZEA  (32*32)      /* Matrix size */
#define GRAY   (64)         /* Maximum number of colors (grayscale) */
#define FLAGS_SIZE (100)

#define FLAGS  0
#define FSIZE  1

#define IX     2
#define IY     3
#define FX     4
#define FY     5

/*
                ------------------------------------------------
--------------------------- Define Image
                ------------------------------------------------
*/

#define IX_H   6
#define IX_C   7

#define IY_H   8
#define IY_C   9

#define IW_D   10
#define IW_D_E 11
#define IW_D_S 12
#define IW_R   13

#define IW_HD  14
#define IW_HD_E 15
#define IW_HD_S 16
#define IW_HR  17

#define IW_CD  18
#define IW_CD_E 19
#define IW_CD_S 20
#define IW_CR  21

/*
                ------------------------------------------------
--------------------------- Define FFT
                ------------------------------------------------
*/

#define FX_H   22
#define FX_C   23

#define FY_H   24
#define FY_C   25
```

```
#define FW_D   26
#define FW_D_E  27
#define FW_D_S  28
#define FW_R   29

#define FW_HD   30
#define FW_HD_E 31
#define FW_HD_S 32
#define FW_HR   33

#define FW_CD   34
#define FW_CD_E 35
#define FW_CD_S 36
#define FW_CR   37

/*
                    ------------------------------------------------
---------------------------- Define general
                    ------------------------------------------------
*/

#define START   38
#define SIZE    39
#define INTPRI  40
#define INTVEC  41
#define ELTIME  42
#define END     43
#define GRY     44

/*
                    ------------------------------------------------
---------------------------- Setup data structures host.dsp
                    ------------------------------------------------
*/

char *symnames[] = {
      "_flags",
      "_flags_size",
      "_ix",
      "_iy",
      "_fx",
      "_fy",
      "_ix_h",          /* image section */
      "_ix_c",
      "_iy_h",
      "_iy_c",
      "_iw_d",
      "_iw_d_e",
      "_iw_d_s",
      "_iw_r",
      "_iw_hd",
      "_iw_hd_e",
      "_iw_hd_s",
      "_iw_hr",
```

```
        "_iw_cd",
        "_iw_cd_e",
        "_iw_cd_s",
        "_iw_cr",
        "_fx_h",            /* fft section  */
        "_fx_c",
        "_fy_h",
        "_fy_c",
        "_fw_d",
        "_fw_d_e",
        "_fw_d_s",
        "_fw_r",
        "_fw_hd",
        "_fw_hd_e",
        "_fw_hd_s",
        "_fw_hr",
        "_fw_cd",
        "_fw_cd_e",
        "_fw_cd_s",
        "_fw_cr",
        "_start_flag",       /* general section */
        "_sizea",
        "_intpri",
        "_intvec",
        "_elapsed_time",
        "_end_flag",
        "_gray",
};

/*
                    ------------------------------------------------
--------------------------- Define global variables
                    ------------------------------------------------
*/

struct symtab   symtab[NUMSYMS];


void c40_handler(void);

time_t start,finish,tsample:    /* setup host timer */
float c_pulse = 1;


/*
                    ------------------------------------------------
--------------------------- Start main program
                    ------------------------------------------------
*/

main()
{

/*
```

```
                    ------------------------------------------------
---------------------------- Define main variables
                    ------------------------------------------------
*/
      int dspid;                /* set DSP id */
      int signum = SIGUSR1;

      u_long entry_address;     /* set entry address reported by DSP */
      struct vc40info hinfo;

      int i,j,k,l, undef = 0;
      u_long sflag;

      float etime;

      u_long temp1 = 0;         /* used for flag dsp confirmation */
      u_long temp2 = 0;


/*
                    ------------------------------------------------
---------------------------- Define main variables (address)
                    ------------------------------------------------
*/

          u_long flags_addr;        /* flags starting address DSP side */
          u_long flags_size_addr;

          u_long ix_addr;           /* ix starting address at DSP side */
          u_long iy_addr;           /* iy starting address at DSP side */
          u_long fx_addr;           /* fx starting address at DSP side */
          u_long fy_addr;           /* fy starting address at DSP side */



      u_long ix_h_addr;         /* image section */
      u_long ix_c_addr;
      u_long iy_h_addr;
      u_long iy_c_addr;

      u_long iw_d_addr;
      u_long iw_d_e_addr;
      u_long iw_d_s_addr;
      u_long iw_r_addr;
      u_long iw_hd_addr;
      u_long iw_hd_e_addr;
      u_long iw_hd_s_addr;
      u_long iw_hr_addr;
      u_long iw_cd_addr;
      u_long iw_cd_e_addr;
      u_long iw_cd_s_addr;
      u_long iw_cr_addr;

      u_long fx_h_addr;         /* fft section */
```

```
    u_long fx_c_addr;
    u_long fy_h_addr;
    u_long fy_c_addr;

    u_long fw_d_addr;
    u_long fw_d_e_addr;
    u_long fw_d_s_addr;
    u_long fw_r_addr;
    u_long fw_hd_addr;
    u_long fw_hd_e_addr;
    u_long fw_hd_s_addr;
    u_long fw_hr_addr;
    u_long fw_cd_addr;
    u_long fw_cd_e_addr;
    u_long fw_cd_s_addr;
    u_long fw_cr_addr;



/*
                ------------------------------------------------
--------------------------- Define main variables (host arrays)
                ------------------------------------------------
*/

    long ix_host[SIZEA];        /* ix array host side */
    long iy_host[SIZEA];        /* iy array host side */
    long fx_host[SIZEA];        /* fx array host side */
    long fy_host[SIZEA];        /* fy array host side */

    long flags_host[FLAGS_SIZE];


    long ix_h_host[GRAY];           /* image section */
    long ix_c_host[GRAY];
    long iy_h_host[GRAY];
    long iy_c_host[GRAY];

    long iw_d_host[SIZEA];
    long iw_d_e_host[SIZEA];
    long iw_d_s_host[SIZEA];
    long iw_r_host[SIZEA];
    long iw_hd_host[GRAY];
    long iw_hd_e_host[GRAY];
    long iw_hd_s_host[GRAY];
    long iw_hr_host[GRAY];
    long iw_cd_host[GRAY];
    long iw_cd_e_host[GRAY];
    long iw_cd_s_host[GRAY];
    long iw_cr_host[GRAY];

    long fx_h_host[GRAY];           /* fft section */
    long fx_c_host[GRAY];
    long fy_h_host[GRAY];
```

```
        long fy_c_host[GRAY];

        long fw_d_host[SIZEA];
        long fw_d_e_host[SIZEA];
        long fw_d_s_host[SIZEA];
        long fw_r_host[SIZEA];
        long fw_hd_host[GRAY];
        long fw_hd_e_host[GRAY];
        long fw_hd_s_host[GRAY];
        long fw_hr_host[GRAY];
        long fw_cd_host[GRAY];
        long fw_cd_e_host[GRAY];
        long fw_cd_s_host[GRAY];
        long fw_cr_host[GRAY];

        printf("\ttemp1 = %d\n ", temp1);
        printf("\ttemp2 = %d\n ", temp2);


/*
                        ------------------------------------------------
--------------------------- DSP1
                        ------------------------------------------------
*/
/*
                        ------------------------------------------------
--------------------------- Open DSP1
                        ------------------------------------------------
*/
start = time(NULL);
tsample = time(NULL);

/* ----------this routine samples the time in seconds----------------- */
printf("\n------------------> Open DSP1:%d seconds\n\n",time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------- */


        dspid = open("/dev/vc40a1", O_RDWR);
        ioctl(dspid, VC40HALT);
        ioctl(dspid, VC40GETINFO, &hinfo);

/*
                        ------------------------------------------------
--------------------------- Load dsp1 program
                        ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------------- */
printf("\n------------------> Load DSP1 program:%d seconds\n\n",\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------- */
```

```
        if (c40_load(dspid, "dsp.x40", &entry_address, NUMSYMS,
                symnames, symtab) == 0) {
            printf("DSP1 coffer is: %s\n", cofferr);
            exit(1);
        }
        else
        printf("\tDSP1 entry address:\t%lx H\n", entry_address);

/*
                    ------------------------------------------------
--------------------------- Symbol checking for DSP1
                    ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------- */
printf("\n------------------> Symbol checking for DSP1:%d seconds\n\n",\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine--------------------------- */


        for (i=0; i<NUMSYMS; i++) {
            if (symtab[i].type == T_UNDEF) {
                printf("DSP1 Symbol %s is undefined!\n", symnames[i]);
                undef = 1;
            }
        }
        if (undef) exit(1);


/*
                    ------------------------------------------------
--------------------------- Write variables to DSP1
                    ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------- */
printf("\n------------------> Write variables to DSP1:%d seconds\n\n",
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine--------------------------- */



        c40_put_long(dspid, symtab[START].val.l, 0L);
        c40_get_long(dspid, symtab[START].val.l, &temp1);
        printf("\tDSP1 initial start flag:\t%x H\n", temp1);

        c40_put_long(dspid, symtab[END].val.l, 0L);
        c40_get_long(dspid, symtab[END].val.l, &temp2);
        printf("\tDSP1 initial end flag:\t%x H\n", temp2);

        c40_put_long(dspid, symtab[INTPRI].val.l, hinfo.intpri);
        c40_put_long(dspid, symtab[INTVEC].val.l, hinfo.intvec);

        signal(signum, c40_handler);
```

```
    ioctl(dspid, VC40ENINT, &signum);

/*
                ------------------------------------------------
-------------------------- Run DSP 1
                ------------------------------------------------
*/
    c40_run(dspid, entry_address);  /* start DSP1 */'



    c40_put_long(dspid, symtab[SIZE].val.l, SIZEA);
    printf("\tDSP1 matrix array size:\t%d\n", SIZEA);

    c40_put_long(dspid, symtab[GRY].val.l, GRAY);
    printf("\tDSP1 gray array size:\t%d\n", GRAY);

    c40_put_long(dspid, symtab[FSIZE].val.l, FLAGS_SIZE);
    printf("\tFlags  array size:\t%d\n", FLAGS_SIZE);

    c40_get_long(dspid, symtab[FLAGS].val.l, &flags_addr);
    printf("\tDSP1 flags addr.:\t%x H\n", flags_addr);


    c40_get_long(dspid, symtab[IX].val.l, &ix_addr);
    printf("\tDSP1 ix address:\t%x H\n", ix_addr);
    c40_get_long(dspid, symtab[IX_H].val.l, &ix_h_addr);
    printf("\tDSP1 ix_h address:\t%x H\n", ix_h_addr);
    c40_get_long(dspid, symtab[IX_C].val.l, &ix_c_addr);
    printf("\tDSP1 ix_c address:\t%x H\n", ix_c_addr);

    c40_get_long(dspid, symtab[IY].val.l, &iy_addr);
    printf("\tDSP1 iy address:\t%x H\n", iy_addr);
    c40_get_long(dspid, symtab[IY_H].val.l, &iy_h_addr);
    printf("\tDSP1 iy_h address:\t%x H\n", iy_h_addr);
    c40_get_long(dspid, symtab[IY_C].val.l, &iy_c_addr);
    printf("\tDSP1 iy_c address:\t%x H\n", iy_c_addr);

    c40_get_long(dspid, symtab[IW_D].val.l, &iw_d_addr);
    printf("\tDSP1 iw_d address:\t%x H\n", iw_d_addr);
    c40_get_long(dspid, symtab[IW_D_E].val.l, &iw_d_e_addr);
    printf("\tDSP1 iw_d_e address:\t%x H\n", iw_d_e_addr);
    c40_get_long(dspid, symtab[IW_D_S].val.l, &iw_d_s_addr);
    printf("\tDSP1 iw_d_s address:\t%x H\n", iw_d_s_addr);
    c40_get_long(dspid, symtab[IW_R].val.l, &iw_r_addr);
    printf("\tDSP1 iw_r address:\t%x H\n", iw_r_addr);

    c40_get_long(dspid, symtab[IW_HD].val.l, &iw_hd_addr);
    printf("\tDSP1 iw_hd address:\t%x H\n", iw_hd_addr);
    c40_get_long(dspid, symtab[IW_HD_E].val.l, &iw_hd_e_addr);
    printf("\tDSP1 iw_hd_e address:\t%x H\n", iw_hd_e_addr);
    c40_get_long(dspid, symtab[IW_HD_S].val.l, &iw_hd_s_addr);
    printf("\tDSP1 iw_hd_s address:\t%x H\n", iw_hd_s_addr);
```

```
c40_get_long(dspid, symtab[IW_HR].val.l, &iw_hr_addr);
printf("\tDSP1 iw_hr address:\t%x H\n", iw_hr_addr);

c40_get_long(dspid, symtab[IW_CD].val.l, &iw_cd_addr);
printf("\tDSP1 iw_cd address:\t%x H\n", iw_cd_addr);
c40_get_long(dspid, symtab[IW_CD_E].val.l, &iw_cd_e_addr);
printf("\tDSP1 iw_cd_e address:\t%x H\n", iw_cd_e_addr);
c40_get_long(dspid, symtab[IW_CD_S].val.l, &iw_cd_s_addr);
printf("\tDSP1 iw_cd_s address:\t%x H\n", iw_cd_s_addr);
c40_get_long(dspid, symtab[IW_CR].val.l, &iw_cr_addr);
printf("\tDSP1 iw_cr address:\t%x H\n", iw_cr_addr);

c40_get_long(dspid, symtab[FX].val.l, &fx_addr);
printf("\tDSP1 fx address:\t%x H\n", fx_addr);
c40_get_long(dspid, symtab[FX_H].val.l, &fx_h_addr);
printf("\tDSP1 fx_h address:\t%x H\n", fx_h_addr);
c40_get_long(dspid, symtab[FX_C].val.l, &fx_c_addr);
printf("\tDSP1 fx_c address:\t%x H\n", fx_c_addr);

c40_get_long(dspid, symtab[FY].val.l, &fy_addr);
printf("\tDSP1 fy address:\t%x H\n", fy_addr);
c40_get_long(dspid, symtab[FY_H].val.l, &fy_h_addr);
printf("\tDSP1 fy_h address:\t%x H\n", fy_h_addr);
c40_get_long(dspid, symtab[FY_C].val.l, &fy_c_addr);
printf("\tDSP1 fy_c address:\t%x H\n", fy_c_addr);

c40_get_long(dspid, symtab[FW_D].val.l, &fw_d_addr);
printf("\tDSP1 fw_d address:\t%x H\n", fw_d_addr);
c40_get_long(dspid, symtab[FW_D_E].val.l, &fw_d_e_addr);
printf("\tDSP1 fw_d_e address:\t%x H\n", fw_d_e_addr);
c40_get_long(dspid, symtab[FW_D_S].val.l, &fw_d_s_addr);
printf("\tDSP1 fw_d_s address:\t%x H\n", fw_d_s_addr);
c40_get_long(dspid, symtab[FW_R].val.l, &fw_r_addr);
printf("\tDSP1 fw_r address:\t%x H\n", fw_r_addr);

c40_get_long(dspid, symtab[FW_HD].val.l, &fw_hd_addr);
printf("\tDSP1 fw_hd address:\t%x H\n", fw_hd_addr);
c40_get_long(dspid, symtab[FW_HD_E].val.l, &fw_hd_e_addr);
printf("\tDSP1 fw_hd_e address:\t%x H\n", fw_hd_e_addr);
c40_get_long(dspid, symtab[FW_HD_S].val.l, &fw_hd_s_addr);
printf("\tDSP1 fw_hd_s address:\t%x H\n", fw_hd_s_addr);
c40_get_long(dspid, symtab[FW_HR].val.l, &fw_hr_addr);
printf("\tDSP1 fw_hr address:\t%x H\n", fw_hr_addr);

c40_get_long(dspid, symtab[FW_CD].val.l, &fw_cd_addr);
printf("\tDSP1 fw_cd address:\t%x H\n", fw_cd_addr);
c40_get_long(dspid, symtab[FW_CD_E].val.l, &fw_cd_e_addr);
printf("\tDSP1 fw_cd_e address:\t%x H\n", fw_cd_e_addr);
c40_get_long(dspid, symtab[FW_CD_S].val.l, &fw_cd_s_addr);
printf("\tDSP1 fw_cd_s address:\t%x H\n", fw_cd_s_addr);
c40_get_long(dspid, symtab[FW_CR].val.l, &fw_cr_addr);
printf("\tDSP1 fw_cr address:\t%x H\n", fw_cr_addr);
```

```
/*
                    ------------------------------------------------
-------------------------- Create 0 data arrarys host + dsp.
                    ------------------------------------------------
*/
      for (i=0; i < SIZEA; i++)  {
            ix_host[i] = 8; iy_host[i] = 4;
                  iw_d_host[i] = 0;
                  iw_d_e_host[i] = 0;
                  iw_d_s_host[i] = 0;
                  iw_r_host[i] = 0;

            fx_host[i] = 16; fy_host[i] = 32;
                  fw_d_host[i] = 0;
                  fw_d_e_host[i] = 0;
                  fw_d_s_host[i] = 0;
                  fw_r_host[i] = 0;
      }
      for (i=0; i < GRAY; i++)  {
            ix_h_host[i] = 0; ix_c_host[i] = 0;
            iy_h_host[i] = 0; iy_c_host[i] = 0;
            fx_h_host[i] = 0; fx_c_host[i] = 0;
            fy_h_host[i] = 0; fy_c_host[i] = 0;


            iw_hd_host[i] = 0; iw_hd_e_host[i] = 0; iw_hd_s_host[i] = 0;
                  iw_hr_host[i] = 0;
            iw_cd_host[i] = 0; iw_cd_e_host[i] = 0; iw_cd_s_host[i] = 0;
                  iw_cr_host[i] = 0;

            fw_hd_host[i] = 0; fw_hd_e_host[i] = 0; fw_hd_s_host[i] = 0;
                  fw_hr_host[i] = 0;
            fw_cd_host[i] = 0; fw_cd_e_host[i] = 0; fw_cd_s_host[i] = 0;
                  fw_cr_host[i] = 0;
      }

      for (i=0; i < FLAGS_SIZE; i++)  {
            flags_host[i] = 0;
      }
/*
                    ------------------------------------------------
-------------------------- Write data array to global/local mem of DSP1
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds---------- */
printf("\n------------------> Write data array to mem.:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------ */


      printf("\tInput data to DSP1 :\n");
```

```
    c40_write_long(dspid, ix_addr, ix_host, SIZEA);
    c40_write_long(dspid, iy_addr, iy_host, SIZEA);

    c40_write_long(dspid, fx_addr, fx_host, SIZEA);
    c40_write_long(dspid, fy_addr, fy_host, SIZEA);

    c40_write_long(dspid, ix_h_addr, ix_h_host, GRAY);
    c40_write_long(dspid, iy_h_addr, iy_h_host, GRAY);
    c40_write_long(dspid, ix_c_addr, ix_c_host, GRAY);
    c40_write_long(dspid, iy_c_addr, iy_c_host, GRAY);

    c40_write_long(dspid, iw_d_addr, iw_d_host, SIZEA);
    c40_write_long(dspid, iw_d_e_addr, iw_d_e_host, SIZEA);
    c40_write_long(dspid, iw_d_s_addr, iw_d_s_host, SIZEA);

    c40_write_long(dspid, iw_r_addr, iw_r_host, SIZEA);

    c40_write_long(dspid, iw_hd_addr, iw_hd_host, GRAY);
    c40_write_long(dspid, iw_hd_e_addr, iw_hd_e_host, GRAY);
    c40_write_long(dspid, iw_hd_s_addr, iw_hd_s_host, GRAY);

    c40_write_long(dspid, iw_hr_addr, iw_hr_host, GRAY);

    c40_write_long(dspid, iw_cd_addr, iw_cd_host, GRAY);
    c40_write_long(dspid, iw_cd_e_addr, iw_cd_e_host, GRAY);
    c40_write_long(dspid, iw_cd_s_addr, iw_cd_s_host, GRAY);

    c40_write_long(dspid, iw_cr_addr, iw_cr_host, GRAY);

    c40_write_long(dspid, fx_h_addr, fx_h_host, GRAY);
    c40_write_long(dspid, fy_h_addr, fy_h_host, GRAY);
    c40_write_long(dspid, fx_c_addr, fx_c_host, GRAY);
    c40_write_long(dspid, fy_c_addr, fy_c_host, GRAY);

    c40_write_long(dspid, fw_d_addr, fw_d_host, SIZEA);
    c40_write_long(dspid, fw_d_e_addr, fw_d_e_host, SIZEA);
    c40_write_long(dspid, fw_d_s_addr, fw_d_s_host, SIZEA);

    c40_write_long(dspid, fw_r_addr, fw_r_host, SIZEA);

    c40_write_long(dspid, fw_hd_addr, fw_hd_host, GRAY);
    c40_write_long(dspid, fw_hd_e_addr, fw_hd_e_host, GRAY);
    c40_write_long(dspid, fw_hd_s_addr, fw_hd_s_host, GRAY);

    c40_write_long(dspid, fw_hr_addr, fw_hr_host, GRAY);

    c40_write_long(dspid, fw_cd_addr, fw_cd_host, GRAY);
    c40_write_long(dspid, fw_cd_e_addr, fw_cd_e_host, GRAY);
    c40_write_long(dspid, fw_cd_s_addr, fw_cd_s_host, GRAY);

    c40_write_long(dspid, fw_cr_addr, fw_cr_host, GRAY);

    c40_write_long(dspid, flags_addr, flags_host, FLAGS_SIZE);
/*
```

```
                    ------------------------------------------------
-------------------------- Start DSP1to execute main and confirm flag
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------- */
printf("\n------------------> Start DSP1 and confirm fl..:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ----------------------end of time routine---------------------- */


      c40_put_long(dspid, symtab[START].val.l, 1L);

      c40_get_long(dspid, symtab[START].val.l, &temp1);
      printf("\tDSP1 start flag:\t%x H\n", temp1);

/*
                    ------------------------------------------------
-------------------------Wait for DSP1 to finish
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------- */
printf("\n------------------> Wait for DSP1 to finish:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ----------------------end of time routine---------------------- */


      printf("\nwaiting for dsp1:\n");
      test_flag:

      for (i=0; i < 1000; i++)       /* set host flag sample time */

      c40_get_long(dspid, symtab[END].val.l, &temp2);

      printf("|");
      if (temp2 == 1) printf("\nDSP program ran sucessfully\n\n");
      else goto test_flag;
      printf("\n\tDSP1 end flag:\t%x H\n", temp2);

/* ----------this routine samples the time in seconds----------- */
printf("\n------------------> DSP1 finished job:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ----------------------end of time routine---------------------- */


/*
                    ------------------------------------------------
-------------------------- Read the results from DSP1
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------- */
printf("\n------------------> Read memory (final result):%d seconds\n\n",\
```

```
      time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine---------------------- */


      c40_read_long(dspid, ix_addr, ix_host, SIZEA);
      c40_read_long(dspid, iy_addr, iy_host, SIZEA);

      c40_read_long(dspid, fx_addr, fx_host, SIZEA);
      c40_read_long(dspid, fy_addr, fy_host, SIZEA);

      c40_read_long(dspid, ix_h_addr, ix_h_host, GRAY);
      c40_read_long(dspid, iy_h_addr, iy_h_host, GRAY);
      c40_read_long(dspid, ix_c_addr, ix_c_host, GRAY);
      c40_read_long(dspid, iy_c_addr, iy_c_host, GRAY);

      c40_read_long(dspid, iw_d_addr, iw_d_host, SIZEA);
      c40_read_long(dspid, iw_d_e_addr, iw_d_e_host, SIZEA);
      c40_read_long(dspid, iw_d_s_addr, iw_d_s_host, SIZEA);

      c40_read_long(dspid, iw_r_addr, iw_r_host, SIZEA);

      c40_read_long(dspid, iw_hd_addr, iw_hd_host, GRAY);
      c40_read_long(dspid, iw_hd_e_addr, iw_hd_e_host, GRAY);
      c40_read_long(dspid, iw_hd_s_addr, iw_hd_s_host, GRAY);

      c40_read_long(dspid, iw_hr_addr, iw_hr_host, GRAY);

      c40_read_long(dspid, iw_cd_addr, iw_cd_host, GRAY);
      c40_read_long(dspid, iw_cd_e_addr, iw_cd_e_host, GRAY);
      c40_read_long(dspid, iw_cd_s_addr, iw_cd_s_host, GRAY);

      c40_read_long(dspid, iw_cr_addr, iw_cr_host, GRAY);

      c40_read_long(dspid, fx_h_addr, fx_h_host, GRAY);
      c40_read_long(dspid, fy_h_addr, fy_h_host, GRAY);
      c40_read_long(dspid, fx_c_addr, fx_c_host, GRAY);
      c40_read_long(dspid, fy_c_addr, fy_c_host, GRAY);

      c40_read_long(dspid, fw_d_addr, fw_d_host, SIZEA);
      c40_read_long(dspid, fw_d_e_addr, fw_d_e_host, SIZEA);
      c40_read_long(dspid, fw_d_s_addr, fw_d_s_host, SIZEA);

      c40_read_long(dspid, fw_r_addr, fw_r_host, SIZEA);

      c40_read_long(dspid, fw_hd_addr, fw_hd_host, GRAY);
      c40_read_long(dspid, fw_hd_e_addr, fw_hd_e_host, GRAY);
      c40_read_long(dspid, fw_hd_s_addr, fw_hd_s_host, GRAY);

      c40_read_long(dspid, fw_hr_addr, fw_hr_host, GRAY);

      c40_read_long(dspid, fw_cd_addr, fw_cd_host, GRAY);
      c40_read_long(dspid, fw_cd_e_addr, fw_cd_e_host, GRAY);
      c40_read_long(dspid, fw_cd_s_addr, fw_cd_s_host, GRAY);
```

```
    c40_read_long(dspid, fw_cr_addr, fw_cr_host, GRAY);

    c40_read_long(dspid, flags_addr, flags_host, FLAGS_SIZE);
/*
                    ------------------------------------------------
--------------------------- Print the results from DSP1
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds---------- */
printf("\n------------------> Print :%d seconds\n\n",\
    time(NULL)-tsample);
tsample = time(NULL);
/* -------------------------end of time routine------------------------ */

printf("\n-----------------------------------------------------------------\n");
printf("\nimages x and y,difference and ratio:\n\n");

    for (i=0; i < SIZEA; i=i+(SIZEA/16))  {
    printf("[%05d]:[ix=%05d]_[iy=%05d]_[iw_d=%05d]_[iw_r=%6.3f]\n",\
     i, ix_host[i],iy_host[i],iw_d_host[i],((float)iw_r_host[i])/1000);
    }


printf("\n-----------------------------------------------------------------\n");
printf("\nhistograms of images x and y,difference and ratio:\n\n");

    for (i=0; i < GRAY; i=i+(GRAY/16))  {
    printf("[%05d]:[ix_h=%05d]_[iy_h=%05d]_[iw_hd=%05d]_[iw_hr= %6.3f]\n",\
     i, ix_h_host[i],iy_h_host[i],\
       iw_hd_host[i],((float)iw_hr_host[i])/1000);
    }

printf("\n-----------------------------------------------------------------\n");
printf("\nCDFs of images x and y,difference and ratio:\n\n");

    for (i=0; i < GRAY; i=i+(GRAY/16))  {
    printf("[%05d]:[ix_c=%05d]_[iy_c=%05d]_[iw_cd=%05d]_[iw_cr= %6.3f]\n",\
     i, ix_c_host[i],iy_c_host[i],\
       iw_cd_host[i],((float)iw_cr_host[i])/1000);
    }


printf("\n-----------------------------------------------------------------\n");
printf("\nfft2d of images x and y,difference and ratio:\n\n");

    for (i=0; i < SIZEA; i=i+(SIZEA/16))  {
    printf("[%05d]:[fx=%05d]_[fy=%05d]_[fw_d=%05d]_[fw_r=%6.3f]\n",\
          i, fx_host[i],fy_host[i],fw_d_host[i],\
            ((float)fw_r_host[i])/1000);
    }

printf("\n-----------------------------------------------------------------\n");
printf("\nff2d histograms of images x and y,difference and ratio:\n\n");
```

```
        for (i=0; i < GRAY; i=i+(GRAY/16))  {
        printf("[%05d]:[fx_h=%05d]_[fy_h=%05d]_[fw_hd=%05d]_[fw_hr= %6.3f]\n",\
         i, fx_h_host[i],fy_h_host[i],\
           fw_hd_host[i],((float)fw_hr_host[i])/1000);
        }

printf("\n-----------------------------------------------------------------\n");
printf("\nfft2d CDFs of images x and y,difference and ratio:\n\n");

        for (i=0; i < GRAY; i=i+(GRAY/16))  {
        printf("[%05d]:[fx_c=%05d]_[fy_c=%05d]_[fw_cd=%05d]_[fw_cr= %6.3f]\n",\
         i, fx_c_host[i],fy_c_host[i],\
           fw_cd_host[i],((float)fw_cr_host[i])/1000);
        }

printf("\n-----------------------------------------------------------------\n");
printf("\nconfirmation flags:\n\n");

        for (i=0; i < 100; i=i+5)  {

            for (j=0; j < 5; j=j+1)  {
            printf("<[%03d]:[%01d]>  ",i+j,flags_host[i+j]);
                    }

printf("\n-----------------------------------------------------------------\n");
        }



 *
                    ------------------------------------------------
--------------------------- Get and print the execution time of DSP1
                    ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------------- */
printf("\n------------------> Print time information:%d seconds\n\n",\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------ */




        c40_get_dsp_float(dspid, symtab[ELTIME].val.l, &etime);
        printf("\n\n\telapsed time dsp-1 : %f usec\n", 1e6*etime);
finish = time(NULL);
        printf("\n\n\t start time : %d  \n",start);
        printf("\t finish time : %d \n",finish);
        printf("\n\telapsed Total time host side: %d seconds(+/- 1 sec) \n",\
            (finish - start));


        return(0);
```

```
}



/*
                ------------------------------------------------
------------------------- Interrupt handling routine
                ------------------------------------------------
*/
void  c40_handler()
{
      printf("\nGot signal from DSP\n\a");

}
```

DSP Program Code:

```
/*
                ************************************************
*************************** Single processor Histogram and CDF (DSP1)
                ************************************************
*/

#include <math.h>
#include <stdlib.h>
#include "/usr/local/hydra/include/hydra.h"

/*  #include "/usr/local/hydra_2.0/axdl/adxl.h" */

/*
                ------------------------------------------------
--------------------------- Define
                ------------------------------------------------
*/
#define  RAMBLK0      0x2ff800
#define  RAMBLK1      0x2ffc00
#define  NEXT         RAMBLK0+256

#define SHARED_ADDR    0xc0000000     /* starting address    */
#define SHARED_SIZE    0xf4240        /* 1000K               */
#define SHARED_BLOCK   0x5dc          /*  1.5K               */

/*
                ------------------------------------------------
--------------------------- Define timer macros
                ------------------------------------------------
*/
```

```
#define ELAPSED_TIME( start, end )     (((end) - (start))*0.0000001)
#define GET_TIMER    (*(unsigned long *)0x00100024)
#define RESET_TIMER   (*(unsigned long *)0x00100020 |= 960)
#define SET_PERIOD(X) (*(unsigned long *)0x00100028 = (unsigned long) X)

#define IX_FLAG     2
#define IY_FLAG     3
#define FX_FLAG     4
#define FY_FLAG     5

#define IX_H_FLAG   6
#define IX_C_FLAG   7

#define IY_H_FLAG   8
#define IY_C_FLAG   9

#define IW_D_FLAG   10
#define IW_D_E_FLAG  11
#define IW_D_S_FLAG  12
#define IW_R_FLAG   13

#define IW_HD_FLAG   14
#define IW_HD_E_FLAG 15
#define IW_HD_S_FLAG 16
#define IW_HR_FLAG   17

#define IW_CD_FLAG   18
#define IW_CD_E_FLAG 19
#define IW_CD_S_FLAG 20
#define IW_CR_FLAG   21

#define FX_H_FLAG   22
#define FX_C_FLAG   23

#define FY_H_FLAG   24
#define FY_C_FLAG   25

#define FW_D_FLAG   26
#define FW_D_E_FLAG  27
#define FW_D_S_FLAG  28
#define FW_R_FLAG   29

#define FW_HD_FLAG   30
#define FW_HD_E_FLAG 31
#define FW_HD_S_FLAG 32
#define FW_HR_FLAG   33

#define FW_CD_FLAG   34
#define FW_CD_E_FLAG 35
#define FW_CD_S_FLAG 36
#define FW_CR_FLAG   37
```

```
/*
                    ------------------------------------------------
--------------------------- Define local and global variables
                    ------------------------------------------------
*/

signed long  *flags = ( long *)(SHARED_ADDR);
unsigned long flags_size;

signed long  *ix = ( long *)(SHARED_ADDR+SHARED_BLOCK*2);
signed long  *iy = ( long *)(SHARED_ADDR+SHARED_BLOCK*3);

signed long  *fx = ( long *)(SHARED_ADDR+SHARED_BLOCK*4);
signed long  *fy = ( long *)(SHARED_ADDR+SHARED_BLOCK*5);
signed long *ix_h = (long *)(SHARED_ADDR+SHARED_BLOCK*6); /* image section */
signed long *ix_c = (long *)(SHARED_ADDR+SHARED_BLOCK*7);

signed long *iy_h = (long *)(SHARED_ADDR+SHARED_BLOCK*8);
signed long *iy_c = (long *)(SHARED_ADDR+SHARED_BLOCK*9);

signed long *iw_d = (long *)(SHARED_ADDR+SHARED_BLOCK*10);
signed long *iw_d_e = (long *)(SHARED_ADDR+SHARED_BLOCK*11);
signed long *iw_d_s = (long *)(SHARED_ADDR+SHARED_BLOCK*12);

signed long *iw_r = (long *)(SHARED_ADDR+SHARED_BLOCK*13);

signed long *iw_hd = (long *)(SHARED_ADDR+SHARED_BLOCK*14);
signed long *iw_hd_e = (long *)(SHARED_ADDR+SHARED_BLOCK*15);
signed long *iw_hd_s = (long *)(SHARED_ADDR+SHARED_BLOCK*16);

signed long *iw_hr = (long *)(SHARED_ADDR+SHARED_BLOCK*17);

signed long *iw_cd = (long *)(SHARED_ADDR+SHARED_BLOCK*18);
signed long *iw_cd_e = (long *)(SHARED_ADDR+SHARED_BLOCK*19);
signed long *iw_cd_s = (long *)(SHARED_ADDR+SHARED_BLOCK*20);

signed long *iw_cr = (long *)(SHARED_ADDR+SHARED_BLOCK*21);

signed long *fx_h = (long *)(SHARED_ADDR+SHARED_BLOCK*22); /* fft section  */
signed long *fx_c = (long *)(SHARED_ADDR+SHARED_BLOCK*23);

signed long *fy_h = (long *)(SHARED_ADDR+SHARED_BLOCK*24);
signed long *fy_c = (long *)(SHARED_ADDR+SHARED_BLOCK*25);

signed long *fw_d = (long *)(SHARED_ADDR+SHARED_BLOCK*26);
signed long *fw_d_e = (long *)(SHARED_ADDR+SHARED_BLOCK*27);
signed long *fw_d_s = (long *)(SHARED_ADDR+SHARED_BLOCK*28);

signed long *fw_r = (long *)(SHARED_ADDR+SHARED_BLOCK*29);

signed long *fw_hd = (long *)(SHARED_ADDR+SHARED_BLOCK*30);
signed long *fw_hd_e = (long *)(SHARED_ADDR+SHARED_BLOCK*31);
signed long *fw_hd_s = (long *)(SHARED_ADDR+SHARED_BLOCK*32);
```

```
signed long *fw_hr = (long *)(SHARED_ADDR+SHARED_BLOCK*33);

signed long *fw_cd = (long *)(SHARED_ADDR+SHARED_BLOCK*34);
signed long *fw_cd_e = (long *)(SHARED_ADDR+SHARED_BLOCK*35);
signed long *fw_cd_s = (long *)(SHARED_ADDR+SHARED_BLOCK*36);

signed long *fw_cr = (long *)(SHARED_ADDR+SHARED_BLOCK*37);




.
unsigned long start_flag = 0;
unsigned long sizea;

int intpri;
int intvec;
float elapsed_time = 0; /* host will read time when done */

unsigned long end_flag = 0;
unsigned long gray;

/*
                    ------------------------------------------------
--------------------------- Define interrupt variables
                    ------------------------------------------------
*/

unsigned long *VIC_virsr = (unsigned long *) 0xbfff0020;
#define HOST_INTERRUPT()        \
      *(VIC_virsr + intpri) = intvec;       \
      *(VIC_virsr) = ((1 << intpri) + 1);


/*
                    ------------------------------------------------
--------------------------- Start DSP1 main
                    ------------------------------------------------
*/
main()  /* Main DSP side program */
{
/*
                    ------------------------------------------------
--------------------------- Define DSP1 main variables
                    ------------------------------------------------
*/
      int i = 0;
      unsigned long timerStart, timerEnd;

      GIE_ON();
      SET_PERIOD(0xffffffff); /* set timer period */

/*
```

```
                    ------------------------------------------------
---------------------------- Wait for host to start reading (check start_flag)
                    ------------------------------------------------
*/

          while (!start_flag);

/*
                    ------------------------------------------------
---------------------------- Start timing routine (set internal timer)
                    ------------------------------------------------
*/

RESET_TIMER;
timerStart = GET_TIMER;

/*
                    ------------------------------------------------
---------------------------- Start loop main loop routine
                    ------------------------------------------------
*/

          flags[IX_FLAG] = 1;              /* set end flag */
          flags[IY_FLAG] = 1;              /* set end flag */
          flags[FX_FLAG] = 1;              /* set end flag */
          flags[FY_FLAG] = 1;              /* set end flag */
/*
                    ------------------------------------------------
---------------------------- Histogram routine
                    ------------------------------------------------

*/
          for( i=0; i < sizea; i++)
          {
               ix_h[ix[i]] = ix_h[ix[i]]+1 ;
               iy_h[iy[i]] = iy_h[iy[i]]+1 ;

               fx_h[fx[i]] = fx_h[fx[i]]+1 ;
               fy_h[fy[i]] = fy_h[fy[i]]+1 ;
          }

          flags[IX_H_FLAG] = 1;            /* set end flag */
          flags[IY_H_FLAG] = 1;            /* set end flag */
          flags[FX_H_FLAG] = 1;            /* set end flag */
          flags[FY_H_FLAG] = 1;            /* set end flag */

/*
                    ------------------------------------------------
---------------------------- CDF routine
                    ------------------------------------------------
*/
          for( i=1; i < gray; i++)
          {
               ix_c[i] = ix_c[i-1]+ix_h[i];
```

```
            iy_c[i] = iy_c[i-1]+iy_h[i];

            fx_c[i] = fx_c[i-1]+fx_h[i];
            fy_c[i] = fy_c[i-1]+fy_h[i];
        }

        flags[IX_C_FLAG] = 1;           /* set end flag */
        flags[IY_C_FLAG] = 1;           /* set end flag */
        flags[FX_C_FLAG] = 1;           /* set end flag */
        flags[FY_C_FLAG] = 1;           /* set end flag */

/*
                ------------------------------------------------
---------------------------- Difference routine iy-ix
                ------------------------------------------------
*/
        for( i=0; i < sizea; i++)
        {
            iw_d[i] = iy[i]-ix[i];
            fw_d[i] = fy[i]-fx[i];
              }

        for( i=0; i < gray; i++)
        {
            iw_hd[i] = iy_h[i]-ix_h[i];
            iw_cd[i] = iy_c[i]-ix_c[i];

            fw_hd[i] = fy_h[i]-fx_h[i];
            fw_cd[i] = fy_c[i]-fx_c[i];
        }

        flags[IW_D_FLAG] = 1;           /* set end flag */
        flags[IW_HD_FLAG] = 1;                  /* set end flag */
        flags[IW_CD_FLAG] = 1;                  /* set end flag */

        flags[FW_D_FLAG] = 1;           /* set end flag */
        flags[FW_HD_FLAG] = 1;                  /* set end flag */
        flags[FW_CD_FLAG] = 1;                  /* set end flag */
/*
                ------------------------------------------------
---------------------------- Ratio routine iy-ix
                ------------------------------------------------
*/
        for( i=0; i < sizea; i++)
        {
                if (ix[i] == 0) iw_r[i] = -9999;
          else iw_r[i] =(1000 * iy[i])/ix[i];
                if (fx[i] == 0) fw_r[i] = -9999;
          else fw_r[i] =(1000 * fy[i])/fx[i];
                }

        for( i=0; i < gray; i++)
        {
                if (ix_h[i] == 0) iw_hr[i] = -9999;
```

```
            else iw_hr[i] =(1000 * iy_h[i])/ix_h[i];

                if (fx_h[i] == 0) fw_hr[i] = -9999;
            else fw_hr[i] =(1000 * fy_h[i])/fx_h[i];

                if (ix_c[i] == 0) iw_cr[i] = -9999;
            else iw_cr[i] =(1000 * iy_c[i])/ix_c[i];

                if (fx_c[i] == 0) fw_cr[i] = -9999;
            else fw_cr[i] =(1000 * fy_c[i])/fx_c[i];
                }

        flags[IW_R_FLAG] = 1;           /* set end flag */
        flags[IW_HR_FLAG] = 1;                  /* set end flag */
        flags[IW_CR_FLAG] = 1;                  /* set end flag */

        flags[FW_R_FLAG] = 1;           /* set end flag */
        flags[FW_HR_FLAG] = 1;                  /* set end flag */
        flags[FW_CR_FLAG] = 1;                  /* set end flag */
/*
                ------------------------------------------------
--------------------------- Stop timer and calculate elapsed time
                ------------------------------------------------
*/

timerEnd = GET_TIMER;
elapsed_time = ELAPSED_TIME(timerStart,timerEnd);

start_flag =0;  /* clear foe next time */

/*
                ------------------------------------------------
--------------------------- Signal Host using DSP flag
                ------------------------------------------------
*/
        end_flag = 1;
}
```

C.2 EXAMPLE RUN FOR SEQUENTIAL CASE (ONE DSP)

A 32x32 matrix with image a = 1 image b = 2 , fft a =3 and fft b =4 is shown as follows:

```
Script started on Sun Mar  5 01:17:53 1995
/dev/ttyp3: Not owner
gorgona.njit.edu% rundsp
      temp1 = 0
      temp2 = 0

------------------> Open DSP1:0 seconds
```

----------> Load DSP1 program:1 seconds

DSP1 entry address: 40001322 H

----------> Symbol checking for DSP1:0 seconds

----------> Write variables to DSP1:0 seconds

DSP1 initial start flag: 0 H
DSP1 initial end flag: 0 H
DSP1 matrix array size: 1024
DSP1 gray array size: 64
Flags array size: 100
DSP1 flags addr.: c0000000 H
DSP1 ix address: c0000bb8 H
DSP1 ix_h address: c0002328 H
DSP1 ix_c address: c0002904 H
DSP1 iy address: c0001194 H
DSP1 iy_h address: c0002ee0 H
DSP1 iy_c address: c00034bc H
DSP1 iw_d address: c0003a98 H
DSP1 iw_d_e address: c0004074 H
DSP1 iw_d_s address: c0004650 H
DSP1 iw_r address: c0004c2c H
DSP1 iw_hd address: c0005208 H
DSP1 iw_hd_e address: c00057e4 H
DSP1 iw_hd_s address: c0005dc0 H
DSP1 iw_hr address: c000639c H
DSP1 iw_cd address: c0006978 H
DSP1 iw_cd_e address: c0006f54 H
DSP1 iw_cd_s address: c0007530 H
DSP1 iw_cr address: c0007b0c H
DSP1 fx address: c0001770 H
DSP1 fx_h address: c00080e8 H
DSP1 fx_c address: c00086c4 H
DSP1 fy address: c0001d4c H
DSP1 fy_h address: c0008ca0 H
DSP1 fy_c address: c000927c H
DSP1 fw_d address: c0009858 H
DSP1 fw_d_e address: c0009e34 H
DSP1 fw_d_s address: c000a410 H
DSP1 fw_r address: c000a9ec H
DSP1 fw_hd address: c000afc8 H
DSP1 fw_hd_e address: c000b5a4 H
DSP1 fw_hd_s address: c000bb80 H
DSP1 fw_hr address: c000c15c H
DSP1 fw_cd address: c000c738 H
DSP1 fw_cd_e address: c000cd14 H
DSP1 fw_cd_s address: c000d2f0 H
DSP1 fw_cr address: c000d8cc H

----------> Write data array to mem.:1 seconds

```
      Input data to DSP1 :

------------------> Start DSP1 and confirm fl.:0 seconds

      DSP1 start flag:       1 H

------------------> Wait for DSP1 to finish:0 seconds


waiting for dsp1:
|
DSP program ran sucessfully


      DSP1 end flag:  1 H

------------------> DSP1 finished job:1 seconds


------------------> Read memory (final result):0 seconds


------------------> Print :0 seconds


---------------------------------------------------------------

images x and y,difference and ratio:

[00000]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00064]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00128]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00192]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00256]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00320]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00384]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00448]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00512]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00576]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00640]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00704]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00768]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00832]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00896]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00960]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]

---------------------------------------------------------------


histograms of images x and y,difference and ratio:

[00000]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
```

[00004]:[ix_h=00000]_[iy_h=01024]_[iw_hd=01024]_[iw_hr= -9.999]
[00008]:[ix_h=01024]_[iy_h=00000]_[iw_hd=-1024]_[iw_hr= 0.000]
[00012]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00016]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00020]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00024]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00028]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00032]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00036]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00040]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00044]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00048]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00052]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00056]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00060]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]

CDFs of images x and y,difference and ratio:

[00000]:[ix_c=00000]_[iy_c=00000]_[iw_cd=00000]_[iw_cr= -9.999]
[00004]:[ix_c=00000]_[iy_c=01024]_[iw_cd=01024]_[iw_cr= -9.999]
[00008]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00012]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00016]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00020]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00024]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00028]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00032]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00036]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00040]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00044]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00048]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00052]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00056]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00060]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]

---

fft2d of images x and y,difference and ratio:

[00000]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00064]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00128]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00192]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00256]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00320]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00384]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00448]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00512]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00576]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00640]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00704]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00768]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]

[00832]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00896]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00960]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]

---

ff2d histograms of images x and y,difference and ratio:

[00000]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00004]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00008]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00012]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00016]:[fx_h=01024]_[fy_h=00000]_[fw_hd=-1024]_[fw_hr= 0.000]
[00020]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00024]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00028]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00032]:[fx_h=00000]_[fy_h=01024]_[fw_hd=01024]_[fw_hr= -9.999]
[00036]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00040]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00044]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00048]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00052]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00056]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00060]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]

---

fft2d CDFs of images x and y,difference and ratio:

[00000]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00004]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00008]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00012]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00016]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00020]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00024]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00028]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00032]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00036]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00040]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00044]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00048]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00052]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00056]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00060]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]

---

confirmation flags:

<[000]:[0]> <[001]:[0]> <[002]:[1]> <[003]:[1]> <[004]:[1]>

<[005]:[1]> <[006]:[1]> <[007]:[1]> <[008]:[1]> <[009]:[1]>

```
<[010]:[1]> <[011]:[0]> <[012]:[0]> <[013]:[1]> <[014]:[1]>
-------------------------------------------------------------
<[015]:[0]> <[016]:[0]> <[017]:[1]> <[018]:[1]> <[019]:[0]>
-------------------------------------------------------------
<[020]:[0]> <[021]:[1]> <[022]:[1]> <[023]:[1]> <[024]:[1]>
-------------------------------------------------------------
<[025]:[1]> <[026]:[1]> <[027]:[0]> <[028]:[0]> <[029]:[1]>
-------------------------------------------------------------
<[030]:[1]> <[031]:[0]> <[032]:[0]> <[033]:[1]> <[034]:[1]>
-------------------------------------------------------------
<[035]:[0]> <[036]:[0]> <[037]:[1]> <[038]:[0]> <[039]:[0]>
-------------------------------------------------------------
<[040]:[0]> <[041]:[0]> <[042]:[0]> <[043]:[0]> <[044]:[0]>
-------------------------------------------------------------
<[045]:[0]> <[046]:[0]> <[047]:[0]> <[048]:[0]> <[049]:[0]>
-------------------------------------------------------------
<[050]:[0]> <[051]:[0]> <[052]:[0]> <[053]:[0]> <[054]:[0]>
-------------------------------------------------------------
<[055]:[0]> <[056]:[0]> <[057]:[0]> <[058]:[0]> <[059]:[0]>
-------------------------------------------------------------
<[060]:[0]> <[061]:[0]> <[062]:[0]> <[063]:[0]> <[064]:[0]>
-------------------------------------------------------------
<[065]:[0]> <[066]:[0]> <[067]:[0]> <[068]:[0]> <[069]:[0]>
-------------------------------------------------------------
<[070]:[0]> <[071]:[0]> <[072]:[0]> <[073]:[0]> <[074]:[0]>
-------------------------------------------------------------
<[075]:[0]> <[076]:[0]> <[077]:[0]> <[078]:[0]> <[079]:[0]>
-------------------------------------------------------------
<[080]:[0]> <[081]:[0]> <[082]:[0]> <[083]:[0]> <[084]:[0]>
-------------------------------------------------------------
<[085]:[0]> <[086]:[0]> <[087]:[0]> <[088]:[0]> <[089]:[0]>
-------------------------------------------------------------
<[090]:[0]> <[091]:[0]> <[092]:[0]> <[093]:[0]> <[094]:[0]>
-------------------------------------------------------------
<[095]:[0]> <[096]:[0]> <[097]:[0]> <[098]:[0]> <[099]:[0]>
-------------------------------------------------------------

------------------> Print time information:2 seconds


      elapsed time dsp-1 : 12181.399390 usec


       start time : 794384281
       finish time : 794384286

      elapsed Total time host side: 5 seconds(+/- 1 sec)
gorgona.njit.edu% exit
gorgona.njit.edu%
script done on Sun Mar  5 01:18:47 1995

script done on Sun Mar  5 01:18:47 1995
```

C.3 PROGRAM CODE FOR PARALLEL CASE (FOUR DSPs)

Host Program:

```
/*      Host side program  for FOUR DSPs
*/

#include <stdio.h>
#include <math.h>
#include <signal.h>
#include "vc40dsp.h"
#include <time.h>
/*
                ------------------------------------------------
--------------------------
                ------------------------------------------------
*/

/*
                ------------------------------------------------
-------------------------- Define Host Variables
                ------------------------------------------------
*/

#define NUMSYMS       (sizeof(symnames)/sizeof(char *))
#define SIZEA   (32*32)      /* Matrix size */
#define GRAY    (64)         /* Maximum number of colors (grayscale) */
#define FLAGS_SIZE (100)

#define FLAGS   0
#define FSIZE   1

#define IX      2
#define IY      3
#define FX      4
#define FY      5

/*
                ------------------------------------------------
-------------------------- Define Image
                ------------------------------------------------
*/

#define IX_H    6
#define IX_C    7

#define IY_H    8
#define IY_C    9

#define IW_D    10
#define IW_D_E  11
#define IW_D_S  12
#define IW_R    13
```

```
#define IW_HD  14
#define IW_HD_E 15
#define IW_HD_S 16
#define IW_HR  17

#define IW_CD  18
#define IW_CD_E 19
#define IW_CD_S 20
#define IW_CR  21

/*
                ------------------------------------------------
--------------------------- Define FFT
                ------------------------------------------------
*/

#define FX_H   22
#define FX_C   23

#define FY_H   24
#define FY_C   25

#define FW_D   26
#define FW_D_E  27
#define FW_D_S  28
#define FW_R   29

#define FW_HD  30
#define FW_HD_E 31
#define FW_HD_S 32
#define FW_HR  33

#define FW_CD  34
#define FW_CD_E 35
#define FW_CD_S 36
#define FW_CR  37

/*
                ------------------------------------------------
--------------------------- Define general
                ------------------------------------------------
*/

#define START  38
#define SIZE   39
#define INTPRI  40
#define INTVEC  41
#define ELTIME  42
#define END    43
#define GRY    44

/*
                ------------------------------------------------
```

```
---------- Setup data structures host,dsp
                    ------------------------------------------------
*/

char *symnames[] = {
      "_flags",
      "_flags_size",
      "_ix",
      "_iy",
      "_fx",
      "_fy",
      "_ix_h",          /* image section */
      "_ix_c",
      "_iy_h",
      "_iy_c",
      "_iw_d",
      "_iw_d_e",
      "_iw_d_s",
      "_iw_r",
      "_iw_hd",
      "_iw_hd_e",
      "_iw_hd_s",
      "_iw_hr",
      "_iw_cd",
      "_iw_cd_e",
      "_iw_cd_s",
      "_iw_cr",
      "_fx_h",          /* fft section  */
      "_fx_c",
      "_fy_h",
      "_fy_c",
      "_fw_d",
      "_fw_d_e",
      "_fw_d_s",
      "_fw_r",
      "_fw_hd",
      "_fw_hd_e",
      "_fw_hd_s",
      "_fw_hr",
      "_fw_cd",
      "_fw_cd_e",
      "_fw_cd_s",
      "_fw_cr",
      "_start_flag",      /* general section */
      "_sizea",
      "_intpri",
      "_intvec",
      "_elapsed_time",
      "_end_flag",
      "_gray",
};

/*
                    ------------------------------------------------
```

```
---------- Define global variables
                    ------------------------------------------------
*/

struct symtab symtab[NUMSYMS],symtab1[NUMSYMS],\
        symtab2[NUMSYMS],symtab3[NUMSYMS];


void c40_handler(void);

time_t start,finish,tsample;    /* setup host timer */
float c_pulse = 1;
.

/*
                    ------------------------------------------------
---------- Start main program
                    ------------------------------------------------
*/

main()
{

/*
                    ------------------------------------------------
---------- Define main variables
                    ------------------------------------------------
*/
      int dspid,dspid1,dspid2,dspid3;              /* set DSP id */

      /* set entry address reported by DSPs */
      u_long entry_address,entry_address1,entry_address2,entry_address3;

          int signum = SIGUSR1, signum1 = SIGUSR2;
     /* int signum2 = SIGUSR3, signum3 = SIGUSR4; */

      struct vc40info hinfo,hinfo1,hinfo2,hinfo3;

      int i,j,k,l, undef=0 ,undef1=0, undef2=0, undef3= 0;
      u_long sflag;

      float etime;

      u_long temp1 = 0;          /* used for flag dsp confirnation */
      u_long temp2 = 0;
      u_long temp3 = 0;
      u_long temp4 = 0;
      u_long temp5 = 0;
      u_long temp6 = 0;
      u_long temp7 = 0;
      u_long temp8 = 0;


/*
```

```
                    ------------------------------------------------
---------------------------- Define main variables (address)
                    ------------------------------------------------
*/

        u_long flags_addr;          /* flags starting address DSP side */
        u_long flags_size_addr;

        u_long ix_addr;             /* ix starting address at DSP side */
        u_long iy_addr;             /* iy starting address at DSP side */
        u_long fx_addr;             /* fx starting address at DSP side */
        u_long fy_addr;             /* fy starting address at DSP side */

        u_long ix_addr1;            /* ix starting address at DSP side */
        u_long iy_addr1;            /* iy starting address at DSP side */
        u_long fx_addr1;            /* fx starting address at DSP side */
        u_long fy_addr1;            /* fy starting address at DSP side */


    u_long ix_h_addr;           /* image section */
    u_long ix_c_addr;
    u_long iy_h_addr;
    u_long iy_c_addr;


    u_long iw_d_addr;
    u_long iw_d_e_addr;
    u_long iw_d_s_addr;
    u_long iw_r_addr;
    u_long iw_hd_addr;
    u_long iw_hd_e_addr;
    u_long iw_hd_s_addr;
    u_long iw_hr_addr;
    u_long iw_cd_addr;
    u_long iw_cd_e_addr;
    u_long iw_cd_s_addr;
    u_long iw_cr_addr;

    u_long fx_h_addr;           /* fft section  */
    u_long fx_c_addr;
    u_long fy_h_addr;
    u_long fy_c_addr;

    u_long fw_d_addr;
    u_long fw_d_e_addr;
    u_long fw_d_s_addr;
    u_long fw_r_addr;
    u_long fw_hd_addr;
    u_long fw_hd_e_addr;
    u_long fw_hd_s_addr;
    u_long fw_hr_addr;
    u_long fw_cd_addr;
    u_long fw_cd_e_addr;
    u_long fw_cd_s_addr;
```

```
    u_long fw_cr_addr;


/*
                ------------------------------------------------
---------------------------- Define main variables (host arrays)
                ------------------------------------------------
*/

    long ix_host[SIZEA];        /* ix array host side */
    long iy_host[SIZEA];        /* iy array host side */
    long fx_host[SIZEA];        /* fx array host side */
    long fy_host[SIZEA];        /* fy array host side */

    long flags_host[FLAGS_SIZE];


    long ix_h_host[GRAY];           /* image section */
    long ix_c_host[GRAY];
    long iy_h_host[GRAY];
    long iy_c_host[GRAY];

    long iw_d_host[SIZEA];
    long iw_d_e_host[SIZEA];
    long iw_d_s_host[SIZEA];
    long iw_r_host[SIZEA];
    long iw_hd_host[GRAY];
    long iw_hd_e_host[GRAY];
    long iw_hd_s_host[GRAY];
    long iw_hr_host[GRAY];
    long iw_cd_host[GRAY];
    long iw_cd_e_host[GRAY];
    long iw_cd_s_host[GRAY];
    long iw_cr_host[GRAY];

    long fx_h_host[GRAY];           /* fft section */
    long fx_c_host[GRAY];
    long fy_h_host[GRAY];
    long fy_c_host[GRAY];

    long fw_d_host[SIZEA];
    long fw_d_e_host[SIZEA];
    long fw_d_s_host[SIZEA];
    long fw_r_host[SIZEA];
    long fw_hd_host[GRAY];
    long fw_hd_e_host[GRAY];
    long fw_hd_s_host[GRAY];
    long fw_hr_host[GRAY];
    long fw_cd_host[GRAY];
    long fw_cd_e_host[GRAY];
    long fw_cd_s_host[GRAY];
    long fw_cr_host[GRAY];
```

```
        printf("\ttemp1 = %d\n ", temp1);
        printf("\ttemp2 = %d\n ", temp2);


/*
                        ------------------------------------------------
--------------------------- DSPs
                        ------------------------------------------------
*/
/*
                        ------------------------------------------------
--------------------------- Open DSP0,DSP1,DSP2,DSP3
                        ------------------------------------------------
*/
start = time(NULL);
tsample = time(NULL);

/* -----------this routine samples the time in seconds----------------- */
printf("\n------------------> Open DSPs:%d seconds\n\n",time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------ */


        dspid = open("/dev/vc40a1", O_RDWR);
        ioctl(dspid, VC40HALT);
        ioctl(dspid, VC40GETINFO, &hinfo);

        dspid1= open("/dev/vc40a2", O_RDWR);
        ioctl(dspid1, VC40HALT);
        ioctl(dspid1, VC40GETINFO, &hinfo1);

        dspid2= open("/dev/vc40a3", O_RDWR);
        ioctl(dspid2, VC40HALT);
        ioctl(dspid2, VC40GETINFO, &hinfo2);

        dspid3= open("/dev/vc40a4", O_RDWR);
        ioctl(dspid, VC40HALT);
        ioctl(dspid, VC40GETINFO, &hinfo3);
/*
                        ------------------------------------------------
--------------------------- Load DSPs program
                        ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------------- */
printf("\n------------------> Load DSPs program:%d seconds\n\n",\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------ */


        if (c40_load(dspid, "dsp.x40", &entry_address, NUMSYMS,
                symnames, symtab) == 0) {
            printf("DSP  coffer is: %s\n", cofferr);
            exit(1);
```

```
    }
    else
    printf("\tDSP  entry address:\t%lx H\n", entry_address);



    if (c40_load(dspid1, "dsp1.x40", &entry_address1, NUMSYMS,
            symnames, symtab1) == 0) {
        printf("DSP1 coffer is: %s\n", cofferr);
        exit(1);
    }
    else
    printf("\tDSP1 entry address:\t%lx H\n", entry_address1);



    if (c40_load(dspid2, "dsp2.x40", &entry_address2, NUMSYMS,
            symnames, symtab2) == 0) {
        printf("DSP2 coffer is: %s\n", cofferr);
        exit(1);
    }
    else
    printf("\tDSP2 entry address:\t%lx H\n", entry_address2);


    if (c40_load(dspid3, "dsp3.x40", &entry_address3, NUMSYMS,
            symnames, symtab3) == 0) {
        printf("DSP3 coffer is: %s\n", cofferr);
        exit(1);
    }
    else
    printf("\tDSP3 entry address:\t%lx H\n", entry_address3);

/*
                ------------------------------------------------
--------------------------- Symbol checking for DSPs
                ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------------- */
printf("\n------------------> Symbol checking for DSPs:%d seconds\n\n",\
    time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------ */


    for (i=0; i<NUMSYMS; i++) {
        if (symtab[i].type == T_UNDEF) {
            printf("DSP  Symbol %s is undefined!\n", symnames[i]);
            undef = 1;
        }
    }
    if (undef) exit(1);
```

```
        for (i=0; i<NUMSYMS; i++) {
            if (symtab1[i].type == T_UNDEF) {
                printf("DSP1 Symbol %s is undefined!\n", symnames[i]);
                undef1 = 1;
            }
        }
        if (undef1) exit(1);


        for (i=0; i<NUMSYMS; i++) {
            if (symtab2[i].type == T_UNDEF) {
                printf("DSP2 Symbol %s is undefined!\n", symnames[i]);
                undef2 = 1;
            }
        }
        if (undef2) exit(1);


        for (i=0; i<NUMSYMS; i++) {
            if (symtab3[i].type == T_UNDEF) {
                printf("DSP3 Symbol %s is undefined!\n", symnames[i]);
                undef3 = 1;
            }
        }
        if (undef3) exit(1);

/*
                    ------------------------------------------------
--------------------------- Write variables to DSPs
                    ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------------- */
printf("\n------------------> Write variables to DSP1:%d seconds\n\n".\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------- */



        c40_put_long(dspid, symtab[START].val.l, 0L);
        c40_get_long(dspid, symtab[START].val.l, &temp1);
        printf("\tDSP  initial start flag:\t%x H\n", temp1);

        c40_put_long(dspid, symtab[END].val.l, 0L);
        c40_get_long(dspid, symtab[END].val.l, &temp2);
        printf("\tDSP  initial end flag:\t%x H\n", temp2);
/*
        c40_put_long(dspid, symtab[INTPRI].val.l, hinfo.intpri);
        c40_put_long(dspid, symtab[INTVEC].val.l, hinfo.intvec);
*/
        signal(signum, c40_handler);
        ioctl(dspid, VC40ENINT, &signum);
```

```
    c40_put_long(dspid1, symtab1[START].val.l, 0L);
    c40_get_long(dspid1, symtab1[START].val.l, &temp1);
    printf("\tDSP1 initial start flag:\t%x H\n", temp1);

    c40_put_long(dspid1, symtab1[END].val.l, 0L);
    c40_get_long(dspid1, symtab1[END].val.l, &temp2);
    printf("\tDSP1 initial end flag:\t%x H\n", temp2);
/*
    c40_put_long(dspid1, symtab1[INTPRI].val.l, hinfo1.intpri);
    c40_put_long(dspid1, symtab1[INTVEC].val.l, hinfo1.intvec);

    signal(signum1, c40_handler);
    ioctl(dspid1, VC40ENINT, &signum1);
*/

    c40_put_long(dspid2, symtab2[START].val.l, 0L);
    c40_get_long(dspid2, symtab2[START].val.l, &temp1);
    printf("\tDSP2 initial start flag:\t%x H\n", temp1);

    c40_put_long(dspid2, symtab2[END].val.l, 0L);
    c40_get_long(dspid2, symtab2[END].val.l, &temp2);
    printf("\tDSP2 initial end flag:\t%x H\n", temp2);
/*
    c40_put_long(dspid2, symtab1[INTPRI].val.l, hinfo2.intpri);
    c40_put_long(dspid2, symtab1[INTVEC].val.l, hinfo2.intvec);

    signal(signum2, c40_handler);
    ioctl(dspid2, VC40ENINT, &signum2);
*/

    c40_put_long(dspid3, symtab3[START].val.l, 0L);
    c40_get_long(dspid3, symtab3[START].val.l, &temp1);
    printf("\tDSP3 initial start flag:\t%x H\n", temp1);

    c40_put_long(dspid3, symtab3[END].val.l, 0L);
    c40_get_long(dspid3, symtab3[END].val.l, &temp2);
    printf("\tDSP3 initial end flag:\t%x H\n", temp2);
/*
    c40_put_long(dspid3, symtab3[INTPRI].val.l, hinfo3.intpri);
    c40_put_long(dspid3, symtab3[INTVEC].val.l, hinfo3.intvec);

    signal(signum3, c40_handler);
    ioctl(dspid3, VC40ENINT, &signum3);
*/


/*
                ------------------------------------------------
--------------------------- Run DSPs
                ------------------------------------------------
*/
    c40_run(dspid, entry_address); /* start DSP */
    c40_run(dspid1, entry_address1); /* start DSP1 */
    c40_run(dspid2, entry_address2); /* start DSP2 */
```

```
    c40_run(dspid3, entry_address3);  /* start DSP3 */



    c40_put_long(dspid, symtab[SIZE].val.l, SIZEA);
    printf("\tDSP  matrix array size:\t%d\n", SIZEA);
    c40_put_long(dspid1, symtab1[SIZE].val.l, SIZEA);
    printf("\tDSP1 matrix array size:\t%d\n", SIZEA);
    c40_put_long(dspid2, symtab2[SIZE].val.l, SIZEA);
    printf("\tDSP2 matrix array size:\t%d\n", SIZEA);
    c40_put_long(dspid3, symtab3[SIZE].val.l, SIZEA);
    printf("\tDSP3 matrix array size:\t%d\n", SIZEA);

    c40_put_long(dspid, symtab[GRY].val.l, GRAY);
    printf("\tDSP  gray array size:\t%d\n", GRAY);
    c40_put_long(dspid1, symtab1[GRY].val.l, GRAY);
    printf("\tDSP1 gray array size:\t%d\n", GRAY);
    c40_put_long(dspid2, symtab2[GRY].val.l, GRAY);
    printf("\tDSP2 gray array size:\t%d\n", GRAY);
    c40_put_long(dspid3, symtab3[GRY].val.l, GRAY);
    printf("\tDSP3 gray array size:\t%d\n", GRAY);

    c40_put_long(dspid, symtab[FSIZE].val.l, FLAGS_SIZE);
    printf("\tFlags  array size:\t%d\n", FLAGS_SIZE);
    c40_put_long(dspid1, symtab1[FSIZE].val.l, FLAGS_SIZE);
    printf("\tFlags  array size:\t%d\n", FLAGS_SIZE);
    c40_put_long(dspid2, symtab2[FSIZE].val.l, FLAGS_SIZE);
    printf("\tFlags  array size:\t%d\n", FLAGS_SIZE);
    c40_put_long(dspid3, symtab3[FSIZE].val.l, FLAGS_SIZE);
    printf("\tFlags  array size:\t%d\n", FLAGS_SIZE);

    c40_get_long(dspid, symtab[FLAGS].val.l, &flags_addr);
    printf("\tDSP  flags addr.:\t%x H\n", flags_addr);
    c40_get_long(dspid1, symtab1[FLAGS].val.l, &flags_addr);
    printf("\tDSP1 flags addr.:\t%x H\n", flags_addr);
    c40_get_long(dspid2, symtab2[FLAGS].val.l, &flags_addr);
    printf("\tDSP2 flags addr.:\t%x H\n", flags_addr);
    c40_get_long(dspid3, symtab3[FLAGS].val.l, &flags_addr);
    printf("\tDSP3 flags addr.:\t%x H\n", flags_addr);


/*
                ------------------------------------------------
---------------------------- DSP 0
                ------------------------------------------------
*/
    c40_get_long(dspid, symtab[IX].val.l, &ix_addr);
    printf("\tDSP  ix address:\t%x H\n", ix_addr);
    c40_get_long(dspid, symtab[IX_H].val.l, &ix_h_addr);
    printf("\tDSP  ix_h address:\t%x H\n", ix_h_addr);
    c40_get_long(dspid, symtab[IX_C].val.l, &ix_c_addr);
    printf("\tDSP  ix_c address:\t%x H\n", ix_c_addr);
```

```
    c40_get_long(dspid, symtab[IY].val.l, &iy_addr);
    printf("\tDSP  iy address:\t%x H\n", iy_addr);
    c40_get_long(dspid, symtab[IY_H].val.l, &iy_h_addr);
    printf("\tDSP  iy_h address:\t%x H\n", iy_h_addr);
    c40_get_long(dspid, symtab[IY_C].val.l, &iy_c_addr);
    printf("\tDSP  iy_c address:\t%x H\n", iy_c_addr);

/*
                ------------------------------------------------
--------------------------- DSP 1
                ------------------------------------------------
*/
    c40_get_long(dspid1, symtab1[IX].val.l, &ix_addr1);
    printf("\tDSP1 ix address:\t%x H\n", ix_addr1);
    c40_get_long(dspid1, symtab1[IY].val.l, &iy_addr1);
    printf("\tDSP1 iy address:\t%x H\n", iy_addr1);

    c40_get_long(dspid1, symtab1[IW_D].val.l, &iw_d_addr);
    printf("\tDSP1 iw_d address:\t%x H\n", iw_d_addr);
    c40_get_long(dspid1, symtab1[IW_D_E].val.l, &iw_d_e_addr);
    printf("\tDSP1 iw_d_e address:\t%x H\n", iw_d_e_addr);
    c40_get_long(dspid1, symtab1[IW_D_S].val.l, &iw_d_s_addr);
    printf("\tDSP1 iw_d_s address:\t%x H\n", iw_d_s_addr);
    c40_get_long(dspid1, symtab1[IW_R].val.l, &iw_r_addr);
    printf("\tDSP1 iw_r address:\t%x H\n", iw_r_addr);

/*
                ------------------------------------------------
--------------------------- DSP 0
                ------------------------------------------------
*/
    c40_get_long(dspid, symtab[IW_HD].val.l, &iw_hd_addr);
    printf("\tDSP  iw_hd address:\t%x H\n", iw_hd_addr);
    c40_get_long(dspid, symtab[IW_HD_E].val.l, &iw_hd_e_addr);
    printf("\tDSP  iw_hd_e address:\t%x H\n", iw_hd_e_addr);
    c40_get_long(dspid, symtab[IW_HD_S].val.l, &iw_hd_s_addr);
    printf("\tDSP  iw_hd_s address:\t%x H\n", iw_hd_s_addr);
    c40_get_long(dspid, symtab[IW_HR].val.l, &iw_hr_addr);
    printf("\tDSP  iw_hr address:\t%x H\n", iw_hr_addr);

    c40_get_long(dspid, symtab[IW_CD].val.l, &iw_cd_addr);
    printf("\tDSP  iw_cd address:\t%x H\n", iw_cd_addr);
    c40_get_long(dspid, symtab[IW_CD_E].val.l, &iw_cd_e_addr);
    printf("\tDSP  iw_cd_e address:\t%x H\n", iw_cd_e_addr);
    c40_get_long(dspid, symtab[IW_CD_S].val.l, &iw_cd_s_addr);
    printf("\tDSP  iw_cd_s address:\t%x H\n", iw_cd_s_addr);
    c40_get_long(dspid, symtab[IW_CR].val.l, &iw_cr_addr);
    printf("\tDSP  iw_cr address:\t%x H\n", iw_cr_addr);

/*
                ------------------------------------------------
--------------------------- DSP 2
                ------------------------------------------------
*/
```

```
    c40_get_long(dspid2, symtab2[FX].val.l, &fx_addr);
    printf("\tDSP2 fx address:\t%x H\n", fx_addr);
    c40_get_long(dspid2, symtab2[FX_H].val.l, &fx_h_addr);
    printf("\tDSP2 fx_h address:\t%x H\n", fx_h_addr);
    c40_get_long(dspid2, symtab2[FX_C].val.l, &fx_c_addr);
    printf("\tDSP2 fx_c address:\t%x H\n", fx_c_addr);

    c40_get_long(dspid2, symtab2[FY].val.l, &fy_addr);
    printf("\tDSP2 fy address:\t%x H\n", fy_addr);
    c40_get_long(dspid2, symtab2[FY_H].val.l, &fy_h_addr);
    printf("\tDSP2 fy_h address:\t%x H\n", fy_h_addr);
    c40_get_long(dspid2, symtab2[FY_C].val.l, &fy_c_addr);
    printf("\tDSP2 fy_c address:\t%x H\n", fy_c_addr);

/*
                ------------------------------------------------
-------------------------- DSP 3
                ------------------------------------------------
*/
    c40_get_long(dspid3, symtab3[FX].val.l, &fx_addr1);
    printf("\tDSP3 fx address:\t%x H\n", fx_addr1);
    c40_get_long(dspid3, symtab3[FY].val.l, &fy_addr1);
    printf("\tDSP3 fy address:\t%x H\n", fy_addr);

    c40_get_long(dspid3, symtab3[FW_D].val.l, &fw_d_addr);
    printf("\tDSP3 fw_d address:\t%x H\n", fw_d_addr);
    c40_get_long(dspid3, symtab3[FW_D_E].val.l, &fw_d_e_addr);
    printf("\tDSP3 fw_d_e address:\t%x H\n", fw_d_e_addr);
    c40_get_long(dspid3, symtab3[FW_D_S].val.l, &fw_d_s_addr);
    printf("\tDSP3 fw_d_s address:\t%x H\n", fw_d_s_addr);
    c40_get_long(dspid3, symtab3[FW_R].val.l, &fw_r_addr);
    printf("\tDSP3 fw_r address:\t%x H\n", fw_r_addr);

/*
                ------------------------------------------------
-------------------------- DSP 2
                ------------------------------------------------
*/
    c40_get_long(dspid2, symtab2[FW_HD].val.l, &fw_hd_addr);
    printf("\tDSP2 fw_hd address:\t%x H\n", fw_hd_addr);
    c40_get_long(dspid2, symtab2[FW_HD_E].val.l, &fw_hd_e_addr);
    printf("\tDSP2 fw_hd_e address:\t%x H\n", fw_hd_e_addr);
    c40_get_long(dspid2, symtab2[FW_HD_S].val.l, &fw_hd_s_addr);
    printf("\tDSP2 fw_hd_s address:\t%x H\n", fw_hd_s_addr);
    c40_get_long(dspid2, symtab2[FW_HR].val.l, &fw_hr_addr);
    printf("\tDSP2 fw_hr address:\t%x H\n", fw_hr_addr);

    c40_get_long(dspid2, symtab2[FW_CD].val.l, &fw_cd_addr);
    printf("\tDSP2 fw_cd address:\t%x H\n", fw_cd_addr);
    c40_get_long(dspid2, symtab2[FW_CD_E].val.l, &fw_cd_e_addr);
    printf("\tDSP2 fw_cd_e address:\t%x H\n", fw_cd_e_addr);
    c40_get_long(dspid2, symtab2[FW_CD_S].val.l, &fw_cd_s_addr);
    printf("\tDSP2 fw_cd_s address:\t%x H\n", fw_cd_s_addr);
    c40_get_long(dspid2, symtab2[FW_CR].val.l, &fw_cr_addr);
```

```
        printf("\tDSP2 fw_cr address:\t%x H\n", fw_cr_addr);



/*
                        ------------------------------------------------
-------------------------- Create 0 data arrarys host + dsp.
                        ------------------------------------------------
*/
        for (i=0; i < SIZEA; i++)  {
                ix_host[i] = 8; iy_host[i] = 4;
                        iw_d_host[i] = 0;
                        iw_d_e_host[i] = 0;
                        iw_d_s_host[i] = 0;
                        iw_r_host[i] = 0;

                fx_host[i] =16 ; fy_host[i] = 32;
                        fw_d_host[i] = 0;
                        fw_d_e_host[i] = 0;
                        fw_d_s_host[i] = 0;
                        fw_r_host[i] = 0;
        }
        for (i=0; i < GRAY; i++)  {
                ix_h_host[i] = 0; ix_c_host[i] = 0;
                iy_h_host[i] = 0; iy_c_host[i] = 0;
                fx_h_host[i] = 0; fx_c_host[i] = 0;
                fy_h_host[i] = 0; fy_c_host[i] = 0;


                iw_hd_host[i] = 0; iw_hd_e_host[i] = 0; iw_hd_s_host[i] = 0;
                        iw_hr_host[i] = 0;
                iw_cd_host[i] = 0; iw_cd_e_host[i] = 0; iw_cd_s_host[i] = 0;
                        iw_cr_host[i] = 0;

                fw_hd_host[i] = 0; fw_hd_e_host[i] = 0; fw_hd_s_host[i] = 0;
                        fw_hr_host[i] = 0;
                fw_cd_host[i] = 0; fw_cd_e_host[i] = 0; fw_cd_s_host[i] = 0;
                        fw_cr_host[i] = 0;
        }

        for (i=0; i < FLAGS_SIZE; i++)  {
                flags_host[i] = 0;
        }
/*
                        ------------------------------------------------
-------------------------- Write data array to global/local mem of DSP1
                        ------------------------------------------------
*/
/* -----------this routine samples the time in seconds----------------- */
printf("\n------------------> Write data array to mem.:%d seconds\n\n",\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------- */
```

```
    printf("\tInput data to DSPs :\n");

/*
                ------------------------------------------------
------------------------- DSP 0
                ------------------------------------------------
*/
    c40_write_long(dspid, ix_addr, ix_host, SIZEA);
    c40_write_long(dspid, iy_addr, iy_host, SIZEA);
/*
                ------------------------------------------------
------------------------- DSP 2
                ------------------------------------------------
*/

    c40_write_long(dspid2, fx_addr, fx_host, SIZEA);
    c40_write_long(dspid2, fy_addr, fy_host, SIZEA);

/*
                ------------------------------------------------
------------------------- DSP 0
                ------------------------------------------------
*/
    c40_write_long(dspid, ix_h_addr, ix_h_host, GRAY);
    c40_write_long(dspid, iy_h_addr, iy_h_host, GRAY);
    c40_write_long(dspid, ix_c_addr, ix_c_host, GRAY);
    c40_write_long(dspid, iy_c_addr, iy_c_host, GRAY);

/*
                ------------------------------------------------
------------------------- DSP 1
                ------------------------------------------------
*/
    c40_write_long(dspid1, ix_addr1, ix_host, SIZEA);
    c40_write_long(dspid1, iy_addr1, iy_host, SIZEA);

    c40_write_long(dspid1, iw_d_addr, iw_d_host, SIZEA);
    c40_write_long(dspid1, iw_d_e_addr, iw_d_e_host, SIZEA);
    c40_write_long(dspid1, iw_d_s_addr, iw_d_s_host, SIZEA);

    c40_write_long(dspid1, iw_r_addr, iw_r_host, SIZEA);

/*
                ------------------------------------------------
------------------------- DSP 0
                ------------------------------------------------
*/
    c40_write_long(dspid, iw_hd_addr, iw_hd_host, GRAY);
    c40_write_long(dspid, iw_hd_e_addr, iw_hd_e_host, GRAY);
    c40_write_long(dspid, iw_hd_s_addr, iw_hd_s_host, GRAY);

    c40_write_long(dspid, iw_hr_addr, iw_hr_host, GRAY);
```

```
    c40_write_long(dspid, iw_cd_addr, iw_cd_host, GRAY);
    c40_write_long(dspid, iw_cd_e_addr, iw_cd_e_host, GRAY);
    c40_write_long(dspid, iw_cd_s_addr, iw_cd_s_host, GRAY);

    c40_write_long(dspid, iw_cr_addr, iw_cr_host, GRAY);

/*
                ------------------------------------------------
------------------------- DSP 2
                ------------------------------------------------
*/
    c40_write_long(dspid2, fx_h_addr, fx_h_host, GRAY);
    c40_write_long(dspid2, fy_h_addr, fy_h_host, GRAY);
    c40_write_long(dspid2, fx_c_addr, fx_c_host, GRAY);
    c40_write_long(dspid2, fy_c_addr, fy_c_host, GRAY);

/*
                ------------------------------------------------
------------------------- DSP 3
                ------------------------------------------------
*/
    c40_write_long(dspid3, fx_addr1, fx_host, SIZEA);
    c40_write_long(dspid3, fy_addr1, fy_host, SIZEA);

    c40_write_long(dspid3, fw_d_addr, fw_d_host, SIZEA);
    c40_write_long(dspid3, fw_d_e_addr, fw_d_e_host, SIZEA);
    c40_write_long(dspid3, fw_d_s_addr, fw_d_s_host, SIZEA);

    c40_write_long(dspid3, fw_r_addr, fw_r_host, SIZEA);

/*
                ------------------------------------------------
------------------------- DSP 2
                ------------------------------------------------
*/
    c40_write_long(dspid2, fw_hd_addr, fw_hd_host, GRAY);
    c40_write_long(dspid2, fw_hd_e_addr, fw_hd_e_host, GRAY);
    c40_write_long(dspid2, fw_hd_s_addr, fw_hd_s_host, GRAY);

    c40_write_long(dspid2, fw_hr_addr, fw_hr_host, GRAY);

    c40_write_long(dspid2, fw_cd_addr, fw_cd_host, GRAY);
    c40_write_long(dspid2, fw_cd_e_addr, fw_cd_e_host, GRAY);
    c40_write_long(dspid2, fw_cd_s_addr, fw_cd_s_host, GRAY);

    c40_write_long(dspid2, fw_cr_addr, fw_cr_host, GRAY);

/*
                ------------------------------------------------
------------------------- DSP0: write flags to global mem using DSP0
                ------------------------------------------------
*/
    c40_write_long(dspid, flags_addr, flags_host, FLAGS_SIZE);
/*
```

```
                    ------------------------------------------------
--------------------------- Start DSPs to execute main and confirm flag
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------------- */
printf("\n------------------> Start DSPs and confirm fl.:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ----------------------end of time routine----------------------- */


      c40_put_long(dspid, symtab[START].val.l, 1L);
      c40_get_long(dspid, symtab[START].val.l, &temp1);
      printf("\tDSP  start flag:\t%01d\n", temp1);

      c40_put_long(dspid1, symtab1[START].val.l, 1L);
      c40_get_long(dspid1, symtab1[START].val.l, &temp2);
      printf("\tDSP1 start flag:\t%01d\n", temp2);

      c40_put_long(dspid2, symtab2[START].val.l, 1L);
      c40_get_long(dspid2, symtab2[START].val.l, &temp3);
      printf("\tDSP2 start flag:\t%01d\n", temp3);

      c40_put_long(dspid3, symtab3[START].val.l, 1L);
      c40_get_long(dspid3, symtab3[START].val.l, &temp4);
      printf("\tDSP3 start flag:\t%01d\n", temp4);


/*
                    ------------------------------------------------
--------------------------Scan DSPs
                    ------------------------------------------------
*/
          temp1 =0;          /* reset temp var. */
          temp2 =0;
          temp3 =0;
          temp4 =0;

      c40_get_long(dspid, symtab[END].val.l, &temp1);
      c40_get_long(dspid1, symtab1[END].val.l, &temp2);
      c40_get_long(dspid2, symtab2[END].val.l, &temp3);
      c40_get_long(dspid3, symtab3[END].val.l, &temp4);
printf("\nDSPs finished:[DSP0:%01d][DSP1:%01d][DSP2:%01d][DSP3:%01d]\
            total:%01d\n",\
       temp1,temp2,temp3,temp4,(temp1+temp2+temp3+temp4));
/*
                    ------------------------------------------------
--------------------------Wait for DSPs to finish
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------------- */
printf("\n------------------> Wait for DSPs to finish:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
```

```
/* ------------------------end of time routine------------------------- */


      printf("\nwaiting for DSPs :\n");
      test_flag:

      for (i=0; i < 100; i++)       /* set host flag sample time */

      c40_get_long(dspid, symtab[END].val.l, &temp1);
      c40_get_long(dspid1, symtab1[END].val.l, &temp2);
      c40_get_long(dspid2, symtab2[END].val.l, &temp3);
      c40_get_long(dspid3, symtab3[END].val.l, &temp4);

printf("\nDSPs finished:[DSP0:%01d][DSP1:%01d][DSP2:%01d][DSP3:%01d]\
            total:%01d\n",\
       temp1,temp2,temp3,temp4,(temp1+temp2+temp3+temp4));
      if ((temp1+temp2+temp3+temp4) == 4)\
         printf("\nDSP program ran sucessfully\n\n");
      else goto test_flag;

      printf("\n\tDSP  end flag:\t%01d\n", temp1);
      printf("\n\tDSP1 end flag:\t%01d\n", temp2);
      printf("\n\tDSP2 end flag:\t%01d\n", temp3);
      printf("\n\tDSP3 end flag:\t%01d\n", temp4);

/* -----------this routine samples the time in seconds------------------ */
printf("\n------------------> DSP1 finished job:%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------- */


/*
                  ------------------------------------------------
---------------------------- Read the results from DSP1
                  ------------------------------------------------
*/
/* -----------this routine samples the time in seconds------------------ */
printf("\n------------------> Read memory (final result):%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------- */

/*
                  ------------------------------------------------
---------------------------- DSP 0
                  ------------------------------------------------
*/

      c40_read_long(dspid, ix_addr, ix_host, SIZEA);
      c40_read_long(dspid, iy_addr, iy_host, SIZEA);

/*
                  ------------------------------------------------
```

```
---------- DSP 2
                    ------------------------------------------------
*/
      c40_read_long(dspid2, fx_addr, fx_host, SIZEA);
      c40_read_long(dspid2, fy_addr, fy_host, SIZEA);

/*
                    ------------------------------------------------
---------- DSP 0
                    ------------------------------------------------
*/
      c40_read_long(dspid, ix_h_addr, ix_h_host, GRAY);
      c40_read_long(dspid, iy_h_addr, iy_h_host, GRAY);
      c40_read_long(dspid, ix_c_addr, ix_c_host, GRAY);
      c40_read_long(dspid, iy_c_addr, iy_c_host, GRAY);

/*
                    ------------------------------------------------
---------- DSP 1
                    ------------------------------------------------
*/
      c40_read_long(dspid1, iw_d_addr, iw_d_host, SIZEA);
      c40_read_long(dspid1, iw_d_e_addr, iw_d_e_host, SIZEA);
      c40_read_long(dspid1, iw_d_s_addr, iw_d_s_host, SIZEA);

      c40_read_long(dspid1, iw_r_addr, iw_r_host, SIZEA);

/*
                    ------------------------------------------------
---------- DSP 0
                    ------------------------------------------------
*/
      c40_read_long(dspid, iw_hd_addr, iw_hd_host, GRAY);
      c40_read_long(dspid, iw_hd_e_addr, iw_hd_e_host, GRAY);
      c40_read_long(dspid, iw_hd_s_addr, iw_hd_s_host, GRAY);

      c40_read_long(dspid, iw_hr_addr, iw_hr_host, GRAY);

      c40_read_long(dspid, iw_cd_addr, iw_cd_host, GRAY);
      c40_read_long(dspid, iw_cd_e_addr, iw_cd_e_host, GRAY);
      c40_read_long(dspid, iw_cd_s_addr, iw_cd_s_host, GRAY);

      c40_read_long(dspid, iw_cr_addr, iw_cr_host, GRAY);

/*
                    ------------------------------------------------
---------- DSP 2
                    ------------------------------------------------
*/
      c40_read_long(dspid2, fx_h_addr, fx_h_host, GRAY);
      c40_read_long(dspid2, fy_h_addr, fy_h_host, GRAY);
      c40_read_long(dspid2, fx_c_addr, fx_c_host, GRAY);
      c40_read_long(dspid2, fy_c_addr, fy_c_host, GRAY);
```

```
/*
                    ------------------------------------------------
--------------------------- DSP 3
                    ------------------------------------------------
*/
      c40_read_long(dspid3, fw_d_addr, fw_d_host, SIZEA);
      c40_read_long(dspid3, fw_d_e_addr, fw_d_e_host, SIZEA);
      c40_read_long(dspid3, fw_d_s_addr, fw_d_s_host, SIZEA);

      c40_read_long(dspid3, fw_r_addr, fw_r_host, SIZEA);

/*
                    ------------------------------------------------
--------------------------- DSP 2
                    ------------------------------------------------
*/
      c40_read_long(dspid2, fw_hd_addr, fw_hd_host, GRAY);
      c40_read_long(dspid2, fw_hd_e_addr, fw_hd_e_host, GRAY);
      c40_read_long(dspid2, fw_hd_s_addr, fw_hd_s_host, GRAY);

      c40_read_long(dspid2, fw_hr_addr, fw_hr_host, GRAY);

      c40_read_long(dspid2, fw_cd_addr, fw_cd_host, GRAY);
      c40_read_long(dspid2, fw_cd_e_addr, fw_cd_e_host, GRAY);
      c40_read_long(dspid2, fw_cd_s_addr, fw_cd_s_host, GRAY);

      c40_read_long(dspid2, fw_cr_addr, fw_cr_host,GRAY);

/*
                    ------------------------------------------------
--------------------------- DSP 1: read flags using DSP0
                    ------------------------------------------------
*/
      c40_read_long(dspid, flags_addr, flags_host, FLAGS_SIZE);
/*
                    ------------------------------------------------
--------------------------- Print the results from DSPs
                    ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------------- */
printf("\n------------------> Print :%d seconds\n\n",\
      time(NULL)-tsample);
tsample = time(NULL);
/* -------------------------end of time routine------------------------ */

printf("\n------------------------------------------------------------\n");
printf("\nimages x and y,difference and ratio:\n\n");

      for (i=0; i < SIZEA; i=i+(SIZEA/16))  {
      printf("[%05d]:[ix=%05d]_[iy=%05d]_[iw_d=%05d]_[iw_r=%6.3f]\n",\
            i, ix_host[i],iy_host[i],iw_d_host[i],\
               ((float)iw_r_host[i])/1000);
      }
```

```
printf("\n-----------------------------------------------------------------\n");
printf("\nhistograms of images x and y,difference and ratio:\n\n");

      for (i=0; i < GRAY; i=i+(GRAY/16))  {
      printf("[%05d]:[ix_h=%05d]_[iy_h=%05d]_[iw_hd=%05d]_[iw_hr= %6.3f]\n",\
       i, ix_h_host[i],iy_h_host[i],\
         iw_hd_host[i],((float)iw_hr_host[i])/1000);
      }

printf("\n-----------------------------------------------------------------\n");
printf("\nCDFs of images x and y,difference and ratio:\n\n");

      for (i=0; i < GRAY; i=i+(GRAY/16))  {
      printf("[%05d]:[ix_c=%05d]_[iy_c=%05d]_[iw_cd=%05d]_[iw_cr= %6.3f]\n",\
       i, ix_c_host[i],iy_c_host[i],\
         iw_cd_host[i],((float)iw_cr_host[i])/1000);
      }


printf("\n-----------------------------------------------------------------\n");
printf("\nfft2d of images x and y,difference and ratio:\n\n");

      for (i=0; i < SIZEA; i=i+(SIZEA/16))  {
      printf("[%05d]:[fx=%05d]_[fy=%05d]_[fw_d=%05d]_[fw_r=%6.3f]\n",\
            i, fx_host[i],fy_host[i],fw_d_host[i],\
                  ((float)fw_r_host[i])/1000);
      }

printf("\n-----------------------------------------------------------------\n");
printf("\nff2d histograms of images x and y,difference and ratio:\n\n");

      for (i=0; i < GRAY; i=i+(GRAY/16))  {
      printf("[%05d]:[fx_h=%05d]_[fy_h=%05d]_[fw_hd=%05d]_[fw_hr= %6.3f]\n",\
       i, fx_h_host[i],fy_h_host[i],\
         fw_hd_host[i],((float)fw_hr_host[i])/1000);
      }

printf("\n-----------------------------------------------------------------\n");
printf("\nfft2d CDFs of images x and y,difference and ratio:\n\n");

      for (i=0; i < GRAY; i=i+(GRAY/16))  {
      printf("[%05d]:[fx_c=%05d]_[fy_c=%05d]_[fw_cd=%05d]_[fw_cr= %6.3f]\n",\
       i, fx_c_host[i],fy_c_host[i],\
         fw_cd_host[i],((float)fw_cr_host[i])/1000);
      }

printf("\n-----------------------------------------------------------------\n");
printf("\nconfirmation flags:\n\n");

      for (i=0; i < 100; i=i+5)  {

          for (j=0; j < 5; j=j+1)  {
          printf("<[%03d]:[%01d]>  ",i+j,flags_host[i+j]);
```

```
                }

printf("\n-------------------------------------------------------------\n");
        }



/*
                ------------------------------------------------
-------------------------- Get and print the execution time of DSP1
                ------------------------------------------------
*/
/* ----------this routine samples the time in seconds----------------- */
printf("\n------------------> Print time information:%d seconds\n\n",\
        time(NULL)-tsample);
tsample = time(NULL);
/* ------------------------end of time routine------------------------ */




        c40_get_dsp_float(dspid, symtab[ELTIME].val.l, &etime);
        printf("\n\n\telapsed time dsp-0 : %f usec\n", 1e6*etime);
        c40_get_dsp_float(dspid1, symtab1[ELTIME].val.l, &etime);
        printf("\n\n\telapsed time dsp-1 : %f usec\n", 1e6*etime);
        c40_get_dsp_float(dspid2, symtab2[ELTIME].val.l, &etime);
        printf("\n\n\telapsed time dsp-2 : %f usec\n", 1e6*etime);
        c40_get_dsp_float(dspid3, symtab3[ELTIME].val.l, &etime);
        printf("\n\n\telapsed time dsp-3 : %f usec\n", 1e6*etime);

finish = time(NULL);
        printf("\n\n\t start time : %d  \n",start);
        printf("\t finish time : %d \n",finish);
        printf("\n\telapsed Total time host side: %d seconds(+/- 1 sec) \n",\
            (finish - start));


        return(0);
}



/*
                ------------------------------------------------
-------------------------- Interrupt handling routine
                ------------------------------------------------
*/
void  c40_handler()
{
        printf("\nGot signal from DSP\n\a");

}
```

DSP 0 Program Code:

```
/*
                ************************************************
************************** (DSP0)
                ************************************************
*/

#include <math.h>
#include <stdlib.h>
#include "/usr/local/hydra/include/hydra.h"

/*  #include "/usr/local/hydra_2.0/axdl/adxl.h" */

/*
                ------------------------------------------------
-------------------------- Define
                ------------------------------------------------
*/
#define  RAMBLK0      0x2ff800
#define  RAMBLK1      0x2ffc00
#define  NEXT         RAMBLK0+256

#define SHARED_ADDR    0x8d000000    /* starting address    */
#define SHARED_SIZE    0xf4240       /* 1000K               */
#define SHARED_BLOCK   0x5dc         /*  1.5k               */
#define LOCAL_ADDR   0xc0000000    /*                  */
#define LOCAL_ADDR1    0xc0000bb8    /* 3000 apart          */
#define SIZEA              0x5dc
#define GRAY               0x40
/*
                ------------------------------------------------
-------------------------- Define timer macros
                ------------------------------------------------
*/


#define ELAPSED_TIME( start, end )     (((end) - (start))*0.0000001)
#define GET_TIMER    (*(unsigned long *)0x00100024)
#define RESET_TIMER   (*(unsigned long *)0x00100020 |= 960)
#define SET_PERIOD(X) (*(unsigned long *)0x00100028 = (unsigned long) X)

#define IX_FLAG    2
#define IY_FLAG    3
#define FX_FLAG    4
#define FY_FLAG    5

#define IX_H_FLAG   6
#define IX_C_FLAG   7

#define IY_H_FLAG   8
```

```
#define IY_C_FLAG   9

#define IW_D_FLAG   10
#define IW_D_E_FLAG 11
#define IW_D_S_FLAG 12
#define IW_R_FLAG   13

#define IW_HD_FLAG   14
#define IW_HD_E_FLAG 15
#define IW_HD_S_FLAG 16
#define IW_HR_FLAG   17

#define IW_CD_FLAG   18
#define IW_CD_E_FLAG 19
#define IW_CD_S_FLAG 20
#define IW_CR_FLAG   21

#define FX_H_FLAG   22
#define FX_C_FLAG   23

#define FY_H_FLAG   24
#define FY_C_FLAG   25

#define FW_D_FLAG   26
#define FW_D_E_FLAG 27
#define FW_D_S_FLAG 28
#define FW_R_FLAG   29

#define FW_HD_FLAG   30
#define FW_HD_E_FLAG 31
#define FW_HD_S_FLAG 32
#define FW_HR_FLAG   33

#define FW_CD_FLAG   34
#define FW_CD_E_FLAG 35
#define FW_CD_S_FLAG 36
#define FW_CR_FLAG   37


/*
                    ------------------------------------------------
---------------------------- Define local and global variables
                    ------------------------------------------------
*/

signed long  *flags = ( long *)(SHARED_ADDR);
unsigned long flags_size;

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP0,DSP1
                    ------------------------------------------------
*/
signed long  *ix = ( long *)(LOCAL_ADDR+SIZEA*0);
```

```
signed long  *iy = ( long *)(LOCAL_ADDR+SIZEA*1);

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP2,DSP3
                    ------------------------------------------------
*/
signed long  *fx = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *fy = ( long *)(LOCAL_ADDR+SIZEA*1);
/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *ix_h = (long *)(LOCAL_ADDR1+GRAY*1);
signed long *ix_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *iy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *iy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP1
                    ------------------------------------------------
*/
signed long *iw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *iw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *iw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);

signed long *iw_r = (long *)(LOCAL_ADDR+SIZEA*6);

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *iw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *iw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *iw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *iw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *iw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *iw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *iw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *iw_cr = (long *)(LOCAL_ADDR1+GRAY*12);

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fx_h = (long *)(LOCAL_ADDR1+GRAY*1); /* fft section  */
```

```
signed long *fx_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *fy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *fy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP3
                    ------------------------------------------------
*/
signed long *fw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *fw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *fw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);


signed long *fw_r = (long *)(LOCAL_ADDR+SIZEA*6);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *fw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *fw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *fw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *fw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *fw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *fw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *fw_cr = (long *)(LOCAL_ADDR1+GRAY*12);




unsigned long start_flag = 0;
unsigned long sizea;

int intpri;
int intvec;

float elapsed_time = 0; /* host will read time when done */

unsigned long end_flag = 0;
unsigned long gray;

/*
                    ------------------------------------------------
---------------------------- Define interrupt variables
                    ------------------------------------------------
```

```
*/

/*
unsigned long *VIC_virsr = (unsigned long *) 0xbfff0020;
#define HOST_INTERRUPT()
      *(VIC_virsr + intpri) = intvec;
      *(VIC_virsr) = ((1 << intpri) + 1);
*/

/*
                  ------------------------------------------------
-------------------------- Start DSP1 main
                  ------------------------------------------------
*/
main()  /* Main DSP side program */
{
/*
                  ------------------------------------------------
-------------------------- Define DSP1 main variables
                  ------------------------------------------------
*/
      int i = 0;
      unsigned long timerStart, timerEnd;

      GIE_ON();
      SET_PERIOD(0xffffffff); /* set timer period */

/*
                  ------------------------------------------------
-------------------------- Wait for host to start reading (check start_flag)
                  ------------------------------------------------
*/

            while (!start_flag);

/*
                  ------------------------------------------------
-------------------------- Start timing routine (set internal timer)
                  ------------------------------------------------
*/

RESET_TIMER;
timerStart = GET_TIMER;

/*
                  ------------------------------------------------
-------------------------- Start loop main loop routine
                  ------------------------------------------------
*/

            flags[IX_FLAG] = 1;              /* set end flag */
            flags[IY_FLAG] = 1;              /* set end flag */
/*
```

```
                    ------------------------------------------------
---------------------------- Histogram routine
                    ------------------------------------------------

*/
          for( i=0; i < sizea; i++)
          {
               ix_h[ix[i]] = ix_h[ix[i]]+1 ;
               iy_h[iy[i]] = iy_h[iy[i]]+1 ;

          }

          flags[IX_H_FLAG] = 1;              /* set end flag */
          flags[IY_H_FLAG] = 1;              /* set end flag */

/*
                    ------------------------------------------------
---------------------------- CDF routine
                    ------------------------------------------------
*/
          for( i=1; i < gray; i++)
          {
               ix_c[i] = ix_c[i-1]+ix_h[i];
               iy_c[i] = iy_c[i-1]+iy_h[i];

          }

          flags[IX_C_FLAG] = 1;              /* set end flag */
          flags[IY_C_FLAG] = 1;              /* set end flag */

/*
                    ------------------------------------------------
---------------------------- Difference(iy-ix) and Ratio(iy/ix)
                    ------------------------------------------------
*/
          for( i=0; i < gray; i++)
          {
               iw_hd[i] = iy_h[i]-ix_h[i];
               iw_cd[i] = iy_c[i]-ix_c[i];

                  if (ix_h[i] == 0)
             iw_hr[i] = -9999;
              else
                              iw_hr[i] = (1000*iy_h[i])/ix_h[i];

                  if (ix_c[i] == 0)
             iw_cr[i] = -9999;
              else
               iw_cr[i] = (1000*iy_c[i])/ix_c[i];
          }

          flags[IW_HD_FLAG] = 1;                    /* set end flag */
          flags[IW_CD_FLAG] = 1;                    /* set end flag */
```

```
/*
                ------------------------------------------------
------------------------- Stop timer and calculate elapsed time
                ------------------------------------------------
*/

timerEnd = GET_TIMER;
elapsed_time = ELAPSED_TIME(timerStart,timerEnd);

start_flag =0;  /* clear foe next time */

/*
                ------------------------------------------------
------------------------- Signal Host using DSP flag
                ------------------------------------------------
*/
        end_flag = 1;
}
```

DSP 1 Program Code:

```
/*
                ************************************************
************************* (DSP1)
                ************************************************
*/

#include <math.h>
#include <stdlib.h>
#include "/usr/local/hydra/include/hydra.h"

/*  #include "/usr/local/hydra_2.0/axdl/adxl.h" */

/*
                ------------------------------------------------
------------------------- Define
                ------------------------------------------------
*/
#define  RAMBLK0      0x2ff800
#define  RAMBLK1      0x2ffc00
#define  NEXT         RAMBLK0+256

#define SHARED_ADDR    0x8d000000    /* starting address    */
#define SHARED_SIZE    0xf4240       /* 1000K               */
#define SHARED_BLOCK   0x5dc         /*  1024 (32x32)       */
#define LOCAL_ADDR   0xc0000000      /*                     */
#define LOCAL_ADDR1    0xc0000bb8    /* 3000 apart          */
#define SIZEA              0x5dc
#define GRAY               0x40
```

```
/*
                    ------------------------------------------------
------------------------- Define timer macros
                    ------------------------------------------------
*/


#define ELAPSED_TIME( start, end )     (((end) - (start))*0.0000001)
#define GET_TIMER     (*(unsigned long *)0x00100024)
#define RESET_TIMER   (*(unsigned long *)0x00100020 |= 960)
#define SET_PERIOD(X) (*(unsigned long *)0x00100028 = (unsigned long) X)

#define IX_FLAG     2
#define IY_FLAG     3
#define FX_FLAG     4
#define FY_FLAG     5

#define IX_H_FLAG   6
#define IX_C_FLAG   7

#define IY_H_FLAG   8
#define IY_C_FLAG   9

#define IW_D_FLAG   10
#define IW_D_E_FLAG 11
#define IW_D_S_FLAG 12
#define IW_R_FLAG   13

#define IW_HD_FLAG  14
#define IW_HD_E_FLAG 15
#define IW_HD_S_FLAG 16
#define IW_HR_FLAG  17

#define IW_CD_FLAG  18
#define IW_CD_E_FLAG 19
#define IW_CD_S_FLAG 20
#define IW_CR_FLAG  21

#define FX_H_FLAG   22
#define FX_C_FLAG   23

#define FY_H_FLAG   24
#define FY_C_FLAG   25

#define FW_D_FLAG   26
#define FW_D_E_FLAG 27
#define FW_D_S_FLAG 28
#define FW_R_FLAG   29

#define FW_HD_FLAG  30
#define FW_HD_E_FLAG 31
#define FW_HD_S_FLAG 32
#define FW_HR_FLAG  33
```

```
#define FW_CD_FLAG  34
#define FW_CD_E_FLAG 35
#define FW_CD_S_FLAG 36
#define FW_CR_FLAG  37


/*
                    ------------------------------------------------
--------------------------- Define local and global variables
                    ------------------------------------------------
*/

signed long  *flags = ( long *)(SHARED_ADDR);
unsigned long flags_size;

/*
                    ------------------------------------------------
--------------------------- Define local and global variables : DSP0,DSP1
                    ------------------------------------------------
*/
signed long  *ix = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *iy = ( long *)(LOCAL_ADDR+SIZEA*1);

/*
                    ------------------------------------------------
--------------------------- Define local and global variables : DSP2,DSP3
                    ------------------------------------------------
*/
signed long  *fx = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *fy = ( long *)(LOCAL_ADDR+SIZEA*1);
/*
                    ------------------------------------------------
--------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *ix_h = (long *)(LOCAL_ADDR1+GRAY*1);
signed long *ix_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *iy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *iy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
--------------------------- Define local and global variables : DSP1
                    ------------------------------------------------
*/
signed long *iw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *iw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *iw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);

signed long *iw_r = (long *)(LOCAL_ADDR+SIZEA*6);


/*
```

```
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *iw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *iw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *iw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *iw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *iw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *iw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *iw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *iw_cr = (long *)(LOCAL_ADDR1+GRAY*12);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fx_h = (long *)(LOCAL_ADDR1+GRAY*1); /* fft section  */
signed long *fx_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *fy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *fy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP3
                    ------------------------------------------------
*/
signed long *fw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *fw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *fw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);


signed long *fw_r = (long *)(LOCAL_ADDR+SIZEA*6);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *fw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *fw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *fw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *fw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *fw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *fw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);
```

```
signed long *fw_cr = (long *)(LOCAL_ADDR1+GRAY*12);
unsigned long start_flag = 0;
unsigned long sizea;

int intpri;
int intvec;

float elapsed_time = 0; /* host will read time when done */

unsigned long end_flag = 0;
unsigned long gray;

/*
                    ------------------------------------------------
---------------------------- Define interrupt variables
                    ------------------------------------------------
*/

/*
unsigned long *VIC_virsr = (unsigned long *) 0xbfff0020;
#define HOST_INTERRUPT()       \
      *(VIC_virsr + intpri) = intvec;       \
      *(VIC_virsr) = ((1 << intpri) + 1);
*/

/*
                    ------------------------------------------------
---------------------------- Start DSP1 main
                    ------------------------------------------------
*/
main()  /* Main DSP side program */
{
/*
                    ------------------------------------------------
---------------------------- Define DSP1 main variables
                    ------------------------------------------------
*/
      int i = 0;
      unsigned long timerStart, timerEnd;

      GIE_ON();
      SET_PERIOD(0xffffffff); /* set timer period */

/*
                    ------------------------------------------------
---------------------------- Wait for host to start reading (check start_flag)
                    ------------------------------------------------
*/

            while (!start_flag);

/*
                    ------------------------------------------------
---------------------------- Start timing routine (set internal timer)
```

```
                ------------------------------------------------
*/

RESET_TIMER;
timerStart = GET_TIMER;

/*
                ------------------------------------------------
--------------------------- Start loop main loop routine
                ------------------------------------------------
*/

/*
                ------------------------------------------------
--------------------------- Difference and ratio
                ------------------------------------------------
*/
        for( i=0; i < sizea; i++)
        {
            iw_d[i] = (iy[i]-ix[i]);

              if (ix[i] == 0) iw_r[i] = -9999; /* test for div.by.0 */
          else  iw_r[i] = (1000*iy[i])/ix[i];

        }

        flags[IW_D_FLAG] = 1;           /* set end flag */
        flags[IW_R_FLAG] = 1;           /* set end flag */
/*
                ------------------------------------------------
--------------------------- Stop timer and calculate elapsed time
                ------------------------------------------------
*/

timerEnd = GET_TIMER;
elapsed_time = ELAPSED_TIME(timerStart,timerEnd);

start_flag =0;  /* clear foe next time */

/*
                ------------------------------------------------
--------------------------- Signal Host using DSP flag
                ------------------------------------------------
*/
        end_flag = 1;
}
```

DSP 2 Program Code:

```
/*
```

```
                ************************************************
************************** (DSP2)
                ************************************************
*/

#include <math.h>
#include <stdlib.h>
#include "/usr/local/hydra/include/hydra.h"

/*  #include "/usr/local/hydra_2.0/axdl/adxl.h" */

/*
                ------------------------------------------------
-------------------------- Define
                ------------------------------------------------
*/
#define  RAMBLK0      0x2ff800
#define  RAMBLK1      0x2ffc00
#define  NEXT         RAMBLK0+256

#define SHARED_ADDR    0x8d000000     /* starting address    */
#define SHARED_SIZE    0xf4240        /* 1000K               */
#define SHARED_BLOCK   0x5dc          /*   1024 (32x32)      */
#define LOCAL_ADDR   0xc0000000    /*                  */
#define LOCAL_ADDR1    0xc0000bb8     /* 3000 apart          */
#define SIZEA              0x5dc
#define GRAY               0x40
/*
                ------------------------------------------------
-------------------------- Define timer macros
                ------------------------------------------------
*/


#define ELAPSED_TIME( start, end )     (((end) - (start))*0.0000001)
#define GET_TIMER    (*(unsigned long *)0x00100024)
#define RESET_TIMER   (*(unsigned long *)0x00100020 |= 960)
#define SET_PERIOD(X) (*(unsigned long *)0x00100028 = (unsigned long) X)

#define IX_FLAG     2
#define IY_FLAG     3
#define FX_FLAG     4
#define FY_FLAG     5

#define IX_H_FLAG   6
#define IX_C_FLAG   7

#define IY_H_FLAG   8
#define IY_C_FLAG   9

#define IW_D_FLAG   10
#define IW_D_E_FLAG 11
#define IW_D_S_FLAG 12
#define IW_R_FLAG   13
```

```
#define IW_HD_FLAG  14
#define IW_HD_E_FLAG 15
#define IW_HD_S_FLAG 16
#define IW_HR_FLAG  17

#define IW_CD_FLAG  18
#define IW_CD_E_FLAG 19
#define IW_CD_S_FLAG 20
#define IW_CR_FLAG  21

#define FX_H_FLAG   22
#define FX_C_FLAG   23

#define FY_H_FLAG   24
#define FY_C_FLAG   25

#define FW_D_FLAG   26
#define FW_D_E_FLAG  27
#define FW_D_S_FLAG  28
#define FW_R_FLAG   29

#define FW_HD_FLAG  30
#define FW_HD_E_FLAG 31
#define FW_HD_S_FLAG 32
#define FW_HR_FLAG  33

#define FW_CD_FLAG  34
#define FW_CD_E_FLAG 35
#define FW_CD_S_FLAG 36
#define FW_CR_FLAG  37


/*
                    ------------------------------------------------
--------------------------- Define local and global variables
                    ------------------------------------------------
*/

signed long  *flags = ( long *)(SHARED_ADDR);
unsigned long flags_size;

/*
                    ------------------------------------------------
--------------------------- Define local and global variables : DSP0,DSP1
                    ------------------------------------------------
*/
signed long  *ix = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *iy = ( long *)(LOCAL_ADDR+SIZEA*1);

/*
                    ------------------------------------------------
--------------------------- Define local and global variables : DSP2,DSP3
                    ------------------------------------------------
```

```
*/
signed long  *fx = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *fy = ( long *)(LOCAL_ADDR+SIZEA*1);
/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *ix_h = (long *)(LOCAL_ADDR1+GRAY*1);
signed long *ix_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *iy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *iy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP1
                    ------------------------------------------------
*/
signed long *iw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *iw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *iw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);

signed long *iw_r = (long *)(LOCAL_ADDR+SIZEA*6);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *iw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *iw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *iw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *iw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *iw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *iw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *iw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *iw_cr = (long *)(LOCAL_ADDR1+GRAY*12);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fx_h = (long *)(LOCAL_ADDR1+GRAY*1); /* fft section  */
signed long *fx_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *fy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *fy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
```

```
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP3
                    ------------------------------------------------
*/
signed long *fw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *fw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *fw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);


signed long *fw_r = (long *)(LOCAL_ADDR+SIZEA*6);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *fw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *fw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *fw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *fw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *fw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *fw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *fw_cr = (long *)(LOCAL_ADDR1+GRAY*12);




unsigned long start_flag = 0;
unsigned long sizea;

int intpri;
int intvec;

float elapsed_time = 0; /* host will read time when done */

unsigned long end_flag = 0;
unsigned long gray;

/*
                    ------------------------------------------------
---------------------------- Define interrupt variables
                    ------------------------------------------------
*/

/*
unsigned long *VIC_virsr = (unsigned long *) 0xbfff0020;
#define HOST_INTERRUPT()        \
        *(VIC_virsr + intpri) = intvec;         \
```

```
    *(VIC_virsr) = ((1 << intpri) + 1);
*/

/*
                    ------------------------------------------------
---------------------------- Start DSP1 main
                    ------------------------------------------------
*/
main()  /* Main DSP side program */
{
/*
                    ------------------------------------------------
---------------------------- Define DSP1 main variables
                    ------------------------------------------------
*/
    int i = 0;
    unsigned long timerStart, timerEnd;

    GIE_ON();
    SET_PERIOD(0xffffffff); /* set timer period */

/*
                    ------------------------------------------------
---------------------------- Wait for host to start reading (check start_flag)
                    ------------------------------------------------
*/

        while (!start_flag);

/*
                    ------------------------------------------------
---------------------------- Start timing routine (set internal timer)
                    ------------------------------------------------
*/

RESET_TIMER:
timerStart = GET_TIMER;

/*
                    ------------------------------------------------
---------------------------- Start loop main loop routine
                    ------------------------------------------------
*/

        flags[FX_FLAG] = 1;             /* set end flag */
        flags[FY_FLAG] = 1;             /* set end flag */
/*
                    ------------------------------------------------
---------------------------- Histogram routine
                    ------------------------------------------------

*/
        for( i=0; i < sizea; i++)
        {
```

```
            fx_h[fx[i]] = fx_h[fx[i]]+1 ;
            fy_h[fy[i]] = fy_h[fy[i]]+1 ;
        }

        flags[FX_H_FLAG] = 1;           /* set end flag */
        flags[FY_H_FLAG] = 1;           /* set end flag */

/*
                ---------------------------------------------
---------------------- CDF routine
                ---------------------------------------------
*/
        for( i=1; i < gray; i++)
        {

            fx_c[i] = fx_c[i-1]+fx_h[i];
            fy_c[i] = fy_c[i-1]+fy_h[i];
        }

        flags[FX_C_FLAG] = 1;           /* set end flag */
        flags[FY_C_FLAG] = 1;           /* set end flag */

/*
                ---------------------------------------------
---------------------- Difference and ratio
                ---------------------------------------------
*/
        for( i=0; i < gray; i++)
        {

            fw_hd[i] = fy_h[i]-fx_h[i];
            fw_cd[i] = fy_c[i]-fx_c[i];

              if (fx_h[i] == 0) fw_hr[i] = -9999;
              else
                fw_hr[i] = (1000*fy_h[i])/fx_h[i];

              if (fx_c[i] == 0) fw_cr[i] = -9999;
              else
            fw_cr[i] = (1000*fy_c[i])/fx_c[i];
        }


        flags[FW_HD_FLAG] = 1;              /* set end flag */
        flags[FW_CD_FLAG] = 1;              /* set end flag */
        flags[FW_HR_FLAG] = 1;              /* set end flag */
        flags[FW_CR_FLAG] = 1;              /* set end flag */
/*
                ---------------------------------------------
---------------------- Stop timer and calculate elapsed time
                ---------------------------------------------
*/
```

```
timerEnd = GET_TIMER;
elapsed_time = ELAPSED_TIME(timerStart,timerEnd);

start_flag =0;  /* clear foe next time */

/*
                ------------------------------------------------
------------------------- Signal Host using DSP flag
                ------------------------------------------------
*/
          end_flag = 1;
}
```

DSP 3 Program Code:

```
/*
                ************************************************
************************** (DSP3)
                ************************************************
*/

#include <math.h>
#include <stdlib.h>
#include "/usr/local/hydra/include/hydra.h"

/*  #include "/usr/local/hydra_2.0/axdl/adxl.h" */

/*
                ------------------------------------------------
------------------------- Define
                ------------------------------------------------
*/
#define  RAMBLK0      0x2ff800
#define  RAMBLK1      0x2ffc00
#define  NEXT         RAMBLK0+256

#define SHARED_ADDR    0x8d000000     /* starting address    */
#define SHARED_SIZE    0xf4240        /* 1000K               */
#define SHARED_BLOCK   0x5dc          /*  1.5k       */
#define LOCAL_ADDR   0xc0000000    /*                 */
#define LOCAL_ADDR1    0xc0000bb8     /* 3000 apart          */
#define SIZEA              0x5dc
#define GRAY               0x40
/*
                ------------------------------------------------
------------------------- Define timer macros
                ------------------------------------------------
*/
```

```
#define ELAPSED_TIME( start, end )     (((end) - (start))*0.0000001)
#define GET_TIMER    (*(unsigned long *)0x00100024)
#define RESET_TIMER  (*(unsigned long *)0x00100020 |= 960)
#define SET_PERIOD(X) (*(unsigned long *)0x00100028 = (unsigned long) X)

#define IX_FLAG    2
#define IY_FLAG    3
#define FX_FLAG    4
#define FY_FLAG    5

#define IX_H_FLAG   6
#define IX_C_FLAG   7

#define IY_H_FLAG   8
#define IY_C_FLAG   9

#define IW_D_FLAG   10
#define IW_D_E_FLAG 11
#define IW_D_S_FLAG 12
#define IW_R_FLAG   13

#define IW_HD_FLAG  14
#define IW_HD_E_FLAG 15
#define IW_HD_S_FLAG 16
#define IW_HR_FLAG  17

#define IW_CD_FLAG  18
#define IW_CD_E_FLAG 19
#define IW_CD_S_FLAG 20
#define IW_CR_FLAG  21

#define FX_H_FLAG   22
#define FX_C_FLAG   23

#define FY_H_FLAG   24
#define FY_C_FLAG   25

#define FW_D_FLAG   26
#define FW_D_E_FLAG 27
#define FW_D_S_FLAG 28
#define FW_R_FLAG   29

#define FW_HD_FLAG  30
#define FW_HD_E_FLAG 31
#define FW_HD_S_FLAG 32
#define FW_HR_FLAG  33

#define FW_CD_FLAG  34
#define FW_CD_E_FLAG 35
#define FW_CD_S_FLAG 36
#define FW_CR_FLAG  37
```

```
/*
                    ------------------------------------------------
------------------------- Define local and global variables
                    ------------------------------------------------
*/

signed long  *flags = ( long *)(SHARED_ADDR);
unsigned long flags_size;

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP0,DSP1
                    ------------------------------------------------
*/
signed long  *ix = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *iy = ( long *)(LOCAL_ADDR+SIZEA*1);

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP2,DSP3
                    ------------------------------------------------
*/
signed long  *fx = ( long *)(LOCAL_ADDR+SIZEA*0);
signed long  *fy = ( long *)(LOCAL_ADDR+SIZEA*1);
/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *ix_h = (long *)(LOCAL_ADDR1+GRAY*1);
signed long *ix_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *iy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *iy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP1
                    ------------------------------------------------
*/
signed long *iw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *iw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *iw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);

signed long *iw_r = (long *)(LOCAL_ADDR+SIZEA*6);


/*
                    ------------------------------------------------
------------------------- Define local and global variables : DSP0
                    ------------------------------------------------
*/
signed long *iw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *iw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
```

```
signed long *iw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *iw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *iw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *iw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *iw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *iw_cr = (long *)(LOCAL_ADDR1+GRAY*12);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fx_h = (long *)(LOCAL_ADDR1+GRAY*1); /* fft section  */
signed long *fx_c = (long *)(LOCAL_ADDR1+GRAY*2);

signed long *fy_h = (long *)(LOCAL_ADDR1+GRAY*3);
signed long *fy_c = (long *)(LOCAL_ADDR1+GRAY*4);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP3
                    ------------------------------------------------
*/
signed long *fw_d = (long *)(LOCAL_ADDR+SIZEA*3);
signed long *fw_d_e = (long *)(LOCAL_ADDR+SIZEA*4);
signed long *fw_d_s = (long *)(LOCAL_ADDR+SIZEA*5);


signed long *fw_r = (long *)(LOCAL_ADDR+SIZEA*6);

/*
                    ------------------------------------------------
---------------------------- Define local and global variables : DSP2
                    ------------------------------------------------
*/
signed long *fw_hd = (long *)(LOCAL_ADDR1+GRAY*5);
signed long *fw_hd_e = (long *)(LOCAL_ADDR1+GRAY*6);
signed long *fw_hd_s = (long *)(LOCAL_ADDR1+GRAY*7);

signed long *fw_hr = (long *)(LOCAL_ADDR1+GRAY*8);

signed long *fw_cd = (long *)(LOCAL_ADDR1+GRAY*9);
signed long *fw_cd_e = (long *)(LOCAL_ADDR1+GRAY*10);
signed long *fw_cd_s = (long *)(LOCAL_ADDR1+GRAY*11);

signed long *fw_cr = (long *)(LOCAL_ADDR1+GRAY*12);
unsigned long start_flag = 0;
unsigned long sizea;

int intpri;
int intvec;
```

```
float elapsed_time = 0; /* host will read time when done */

unsigned long end_flag = 0;
unsigned long gray;

/*
                    ------------------------------------------------
------------------------- Define interrupt variables
                    ------------------------------------------------
*/

/*
unsigned long *VIC_virsr = (unsigned long *) 0xbfff0020;
#define HOST_INTERRUPT()
      *(VIC_virsr + intpri) = intvec;
      *(VIC_virsr) = ((1 << intpri) + 1);
*/

/*
                    ------------------------------------------------
------------------------- Start DSP1 main
                    ------------------------------------------------
*/
main()  /* Main DSP side program */
{
/*
                    ------------------------------------------------
------------------------- Define DSP1 main variables
                    ------------------------------------------------
*/
      int i = 0;
      unsigned long timerStart, timerEnd;

      GIE_ON();
      SET_PERIOD(0xffffffff); /* set timer period */

/*
                    ------------------------------------------------
------------------------- Wait for host to start reading (check start_flag)
                    ------------------------------------------------
*/

            while (!start_flag);

/*
                    ------------------------------------------------
------------------------- Start timing routine (set internal timer)
                    ------------------------------------------------
*/

RESET_TIMER;
timerStart = GET_TIMER;
```

```
/*
                    ---------------------------------------------
------------------------- Start loop main loop routine
                    ---------------------------------------------
*/
          flags[FX_FLAG] = 1;              /* set end flag */
          flags[FY_FLAG] = 1;              /* set end flag */
/*
                    ---------------------------------------------
------------------------- Difference and ratio routine
                    ---------------------------------------------
*/
          for( i=0; i < sizea; i++)
          {

               fw_d[i] = fy[i]-fx[i];

                 if (fx[i] == 0) fw_r[i] = -9999;
                 else
               fw_r[i] = (1000*fy[i])/fx[i];
          }

          flags[FW_D_FLAG] = 1;            /* set end flag */
          flags[FW_R_FLAG] = 1;            /* set end flag */
/*
                    ---------------------------------------------
------------------------- Stop timer and calculate elapsed time
                    ---------------------------------------------
*/

timerEnd = GET_TIMER;
elapsed_time = ELAPSED_TIME(timerStart,timerEnd);

start_flag =0;  /* clear foe next time */

/*
                    ---------------------------------------------
------------------------- Signal Host using DSP flag
                    ---------------------------------------------
*/
          end_flag = 1;
}
```

C.4 EXAMPLE RUN FOR PARALLEL CASE (FOUR DSPs)

```
Script started on Sun Mar  5 01:21:11 1995
/dev/ttyp3: Not owner
gorgona.njit.edu% rundsp
      temp1 = 0
      temp2 = 0
```

----------> Open DSPs:0 seconds

----------> Load DSPs program:2 seconds

DSP entry address: 4000128b H
DSP1 entry address: 40001240 H
DSP2 entry address: 4000128d H
DSP3 entry address: 40001243 H

----------> Symbol checking for DSPs:0 seconds

----------> Write variables to DSP1:0 seconds

DSP initial start flag: 0 H
DSP initial end flag: 0 H
DSP1 initial start flag: 0 H
DSP1 initial end flag: 0 H
DSP2 initial start flag: 0 H
DSP2 initial end flag: 0 H
DSP3 initial start flag: 0 H
DSP3 initial end flag: 0 H
DSP matrix array size: 1024
DSP1 matrix array size: 1024
DSP2 matrix array size: 1024
DSP3 matrix array size: 1024
DSP gray array size: 64
DSP1 gray array size: 64
DSP2 gray array size: 64
DSP3 gray array size: 64
Flags array size: 100
Flags array size: 100
Flags array size: 100
Flags array size: 100
DSP flags addr.: 8d000000 H
DSP1 flags addr.: 8d000000 H
DSP2 flags addr.: 8d000000 H
DSP3 flags addr.: 8d000000 H
DSP ix address: c0000000 H
DSP ix_h address: c0000bf8 H
DSP ix_c address: c0000c38 H
DSP iy address: c00005dc H
DSP iy_h address: c0000c78 H
DSP iy_c address: c0000cb8 H
DSP1 ix address: c0000000 H
DSP1 iy address: c00005dc H
DSP1 iw_d address: c0001194 H
DSP1 iw_d_e address: c0001770 H
DSP1 iw_d_s address: c0001d4c H
DSP1 iw_r address: c0002328 H
DSP iw_hd address: c0000cf8 H
DSP iw_hd_e address: c0000d38 H
DSP iw_hd_s address: c0000d78 H

```
      DSP  iw_hr address:    c0000db8 H
      DSP  iw_cd address:    c0000df8 H
      DSP  iw_cd_e address:  c0000e38 H
      DSP  iw_cd_s address:  c0000e78 H
      DSP  iw_cr address:    c0000eb8 H
      DSP2 fx address:       c0000000 H
      DSP2 fx_h address:     c0000bf8 H
      DSP2 fx_c address:     c0000c38 H
      DSP2 fy address:       c00005dc H
      DSP2 fy_h address:     c0000c78 H
      DSP2 fy_c address:     c0000cb8 H
      DSP3 fx address:       c0000000 H
      DSP3 fy address:       c00005dc H
      DSP3 fw_d address:     c0001194 H
      DSP3 fw_d_e address:   c0001770 H
      DSP3 fw_d_s address:   c0001d4c H
      DSP3 fw_r address:     c0002328 H
      DSP2 fw_hd address:    c0000cf8 H
      DSP2 fw_hd_e address:  c0000d38 H
      DSP2 fw_hd_s address:  c0000d78 H
      DSP2 fw_hr address:    c0000db8 H
      DSP2 fw_cd address:    c0000df8 H
      DSP2 fw_cd_e address:  c0000e38 H
      DSP2 fw_cd_s address:  c0000e78 H
      DSP2 fw_cr address:    c0000eb8 H

----------> Write data array to mem.:1 seconds

      Input data to DSPs :

----------> Start DSPs and confirm fl.:0 seconds

      DSP  start flag:       1
      DSP1 start flag:       1
      DSP2 start flag:       1
      DSP3 start flag:       1

DSPs finished:[DSP0:1][DSP1:1][DSP2:1][DSP3:0]   total:3

----------> Wait for DSPs to finish:0 seconds


waiting for DSPs :

DSPs finished:[DSP0:1][DSP1:1][DSP2:1][DSP3:1]   total:4

DSP program ran sucessfully


      DSP  end flag:  1

      DSP1 end flag:  1

      DSP2 end flag:  1
```

DSP3 end flag: 1

----------> DSP1 finished job:1 seconds

----------> Read memory (final result):0 seconds

----------> Print :0 seconds

---

images x and y,difference and ratio:

[00000]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00064]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00128]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00192]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00256]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00320]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00384]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00448]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00512]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00576]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00640]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00704]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00768]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00832]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00896]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]
[00960]:[ix=00008]_[iy=00004]_[iw_d=-0004]_[iw_r= 0.500]

---

histograms of images x and y,difference and ratio:

[00000]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00004]:[ix_h=00000]_[iy_h=01024]_[iw_hd=01024]_[iw_hr= -9.999]
[00008]:[ix_h=01024]_[iy_h=00000]_[iw_hd=-1024]_[iw_hr= 0.000]
[00012]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00016]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00020]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00024]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00028]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00032]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00036]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00040]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00044]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00048]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00052]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00056]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]
[00060]:[ix_h=00000]_[iy_h=00000]_[iw_hd=00000]_[iw_hr= -9.999]

CDFs of images x and y,difference and ratio:

[00000]:[ix_c=00000]_[iy_c=00000]_[iw_cd=00000]_[iw_cr= -9.999]
[00004]:[ix_c=00000]_[iy_c=01024]_[iw_cd=01024]_[iw_cr= -9.999]
[00008]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00012]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00016]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00020]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00024]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00028]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00032]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00036]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00040]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00044]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00048]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00052]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00056]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]
[00060]:[ix_c=01024]_[iy_c=01024]_[iw_cd=00000]_[iw_cr= 1.000]

---

fft2d of images x and y,difference and ratio:

[00000]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00064]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00128]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00192]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00256]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00320]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00384]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00448]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00512]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00576]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00640]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00704]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00768]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00832]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00896]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]
[00960]:[fx=00016]_[fy=00032]_[fw_d=00016]_[fw_r= 2.000]

---

ff2d histograms of images x and y,difference and ratio:

[00000]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00004]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00008]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00012]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00016]:[fx_h=01024]_[fy_h=00000]_[fw_hd=-1024]_[fw_hr= 0.000]
[00020]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00024]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00028]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]

[00032]:[fx_h=00000]_[fy_h=01024]_[fw_hd=01024]_[fw_hr= -9.999]
[00036]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00040]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00044]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00048]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00052]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00056]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]
[00060]:[fx_h=00000]_[fy_h=00000]_[fw_hd=00000]_[fw_hr= -9.999]

---

fft2d CDFs of images x and y,difference and ratio:

[00000]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00004]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00008]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00012]:[fx_c=00000]_[fy_c=00000]_[fw_cd=00000]_[fw_cr= -9.999]
[00016]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00020]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00024]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00028]:[fx_c=01024]_[fy_c=00000]_[fw_cd=-1024]_[fw_cr= 0.000]
[00032]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00036]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00040]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00044]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00048]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00052]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00056]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]
[00060]:[fx_c=01024]_[fy_c=01024]_[fw_cd=00000]_[fw_cr= 1.000]

---

confirmation flags:

<[000]:[0]> <[001]:[0]> <[002]:[1]> <[003]:[1]> <[004]:[1]>

<[005]:[1]> <[006]:[1]> <[007]:[1]> <[008]:[1]> <[009]:[1]>

<[010]:[1]> <[011]:[0]> <[012]:[0]> <[013]:[1]> <[014]:[1]>

<[015]:[0]> <[016]:[0]> <[017]:[0]> <[018]:[1]> <[019]:[0]>

<[020]:[0]> <[021]:[0]> <[022]:[1]> <[023]:[1]> <[024]:[1]>

<[025]:[1]> <[026]:[1]> <[027]:[0]> <[028]:[0]> <[029]:[1]>

<[030]:[1]> <[031]:[0]> <[032]:[0]> <[033]:[1]> <[034]:[1]>

<[035]:[0]> <[036]:[0]> <[037]:[1]> <[038]:[0]> <[039]:[0]>

<[040]:[0]> <[041]:[0]> <[042]:[0]> <[043]:[0]> <[044]:[0]>

<[045]:[0]> <[046]:[0]> <[047]:[0]> <[048]:[0]> <[049]:[0]>

```
<[050]:[0]> <[051]:[0]> <[052]:[0]> <[053]:[0]> <[054]:[0]>
------------------------------------------------------------
<[055]:[0]> <[056]:[0]> <[057]:[0]> <[058]:[0]> <[059]:[0]>
------------------------------------------------------------
<[060]:[0]> <[061]:[0]> <[062]:[0]> <[063]:[0]> <[064]:[0]>
------------------------------------------------------------
<[065]:[0]> <[066]:[0]> <[067]:[0]> <[068]:[0]> <[069]:[0]>
------------------------------------------------------------
<[070]:[0]> <[071]:[0]> <[072]:[0]> <[073]:[0]> <[074]:[0]>
------------------------------------------------------------
<[075]:[0]> <[076]:[0]> <[077]:[0]> <[078]:[0]> <[079]:[0]>
------------------------------------------------------------
<[080]:[0]> <[081]:[0]> <[082]:[0]> <[083]:[0]> <[084]:[0]>
------------------------------------------------------------
<[085]:[0]> <[086]:[0]> <[087]:[0]> <[088]:[0]> <[089]:[0]>
------------------------------------------------------------
<[090]:[0]> <[091]:[0]> <[092]:[0]> <[093]:[0]> <[094]:[0]>
------------------------------------------------------------
<[095]:[0]> <[096]:[0]> <[097]:[0]> <[098]:[0]> <[099]:[0]>
------------------------------------------------------------

----------> Print time information:1 seconds


      elapsed time dsp-0 : 2055.499936 usec


      elapsed time dsp-1 : 3851.999994 usec


      elapsed time dsp-2 : 2004.999900 usec


      elapsed time dsp-3 : 3965.700045 usec


       start time : 794384475
       finish time : 794384480

      elapsed Total time host side: 5 seconds(+/- 1 sec)
gorgona.njit.edu% exit
gorgona.njit.edu%
script done on Sun Mar  5 01:21:27 1995
```

REFERENCES

1. R. C. Gonzales and R. E. Woods, *Digital Image Processing*, Addison - Wesley Publishing Company, Reading, Massachusetts , 1992.

2. X.Li, S.G. Ziavras, and C.N. Manikopoulos, "Parallel DSP Algorithms on TurboNet: An Experimental Hybrid Message-Passing/Shared - Memory Architecture," Concurrency: Practice and Experience, to appear.

3. Force Computers Inc. *SPARC CPU-2CE Technical Reference Manual*. April 1993.

4. Ariel Corporation, *User's Manual for the V-C40 Hydra. Version 0.60*, February, 1994.

5. Texas Instruments, *TMS320C4x User's Guide*, 1993.

6. Kai Hwang, *Advanced Computer Architecture with Parallel Programming*, McGraw-Hill, Inc., New York, 1993.

7. T.S. Huang, *Image Sequence Analysis*, Springer-Verlag, New York , NY, 1981.

8. Texas Instruments, *TMS320C4x Floating-Point DSP Optimal C Compiler User's Guide*, 1991.